
FRONT-END
DEVELOPER
HANDBOOK
2017
WAREHOUSE
FACTORY
HOTEL
MUSEUM
Pharmacy
PRESENTED BY:
FRONTEND
MASTERS

目录

前端开发者指南（2017）

掘金翻译计划 沪江Web前端团队



作者：科迪·林黎（Cody Lindley），由「前端大师（Frontend Masters）」倾情赞助。

这是一本可供任何人使用的指南，用于学习前端开发实践。该指南大体上勾勒出了前端工程的轮廓，同时也讨论了前端工程的实践：2017 年，如何学习前端工程，用什么工具来实践？

笔者有意将本书打造为一份专业资料，为想要或正在实践的前端开发者们提供学习材料和开发工具。其次，它同样可供主管、CTO、讲师和猎头们深入探索前端开发实践。

本书内容偏向于 WEB 技术（HTML、CSS、DOM、JavaScript）和以这些技术为根基直接构建而成的开源技术。书中引用和讨论的材料要么就是同类翘楚，要么就是解决问题的流行方案。

本书不是一本囊括所有前端可用资源的综合纲领。其价值在于为恰好够用的分类信息搜罗简洁、聚焦且符合时宜的甄选内容，以免在特别话题下钻了牛角尖。

预期本书每年都迭代一次内容。

本书分为三部分。

第一部分：前端实践

第一部分概述了前端工程实践。

第二部分：学习前端开发

第二部分指出了学习成为一个前端开发者所需的自学资源和教学资源（译者注：教学资源包括有讲师指导的付费课程、计划、学院和训练营）。

第三部分：前端开发工具

第三部分简要地介绍和指出了一些前端圈内的工具。

文章目录

- 模块／包管理工具
- 托管工具
- 项目管理以及代码托管工具
- 协作与沟通工具
- 内容管理 托管／API 工具
- 后端即服务工具
- 离线工具
- 安全工具
- 构建工具
- 部署工具
- 网站／应用监控工具
- JavaScript 错误报告／监控
- 性能工具
- 寻找工具的工具

下载 **.pdf**、**.epub**、或 **.mobi** 格式的电子书:

- 中文：https://www.gitbook.com/book/sqrtthree/front-end-handbook-2017/details
- 英文：https://www.gitbook.com/book/frontendmasters/front-end-handbook-2017/details

贡献内容、提建议或者修复 **GitHub** 上的 **bugs**:

- https://github.com/xitu/front-end-handbook-2017



致谢

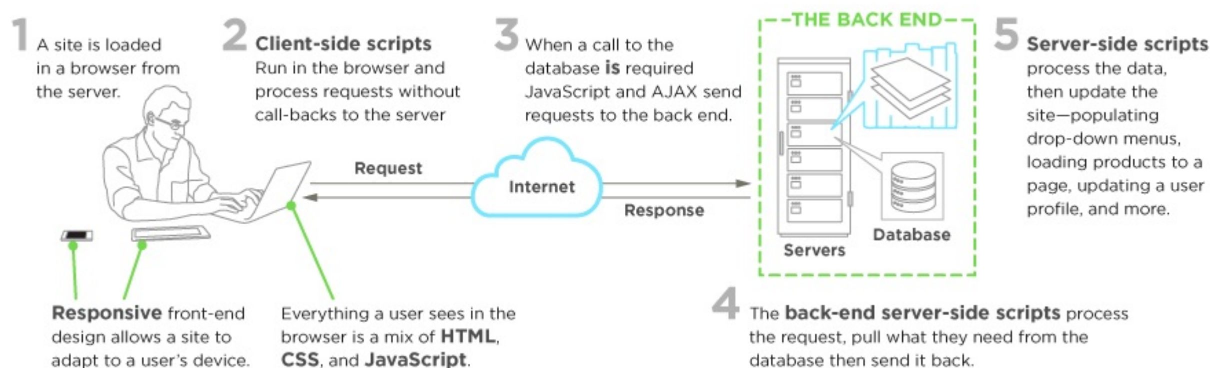
- 沪江Web前端团队
- 掘金翻译计划

前端开发者是什么？

> WEB 前端开发，也作客户端开发，是给网站或网页应用生产 HTML、CSS 和 JavaScript 的过程，它让用户得以浏览网站或网页应用并与之交互。创建网站前端的工具与技术时常变革，因此开发者应对业内发展近况保持清醒，这是前端开发者面对的挑战。
>
> 保证用户打开网站时，以一种易读且关联的形式浏览信息，是设计网站的目的。目前这个需求变得更棘手：如今用户使用的设备种类繁多，设备对应的屏幕尺寸和方案亦然，这迫使设计者在开发站点时考虑这些方面。他们要保证站点在不同的浏览器（跨浏览器），不同的操作系统（跨平台）以及不同的设备（跨设备）上正确运行，这要求开发者缜密地制定方案。
>
> *https://en.wikipedia.org/wiki/Front-end_web_development*

HTML、CSS 和 JavaScript:

前端开发者使用 WEB 技术（例如 HTML、CSS、 DOM 和 JavaScript）来建造网站和开发应用。他们使用 WEB 技术在 WEB 平台或非 WEB 框架（比如 NativeScript 框架）上设计程序。



图片来源：https://www.upwork.com/hiring/development/front-end-developer/

通常前端开发者通过学习 HTML、CSS 和 JS 来入门。HTML、CSS 和 JS 代码在 WEB 浏览器、无头浏览器（Headless Browsers）、WebView 上运行，或者被用于原生运行环境的程序设计。我稍后解释这四种运行场景。

WEB 浏览器

WEB 浏览器用于从万维网（WWW.）检索、呈现和遍历信息。通常浏览器在台式机、笔记本、平板或手机上运行，如今，浏览器可以在几乎所有物体（比如冰箱，汽车）上运行。

最常见的 WEB 浏览器如下（按使用度排序）：

- Chrome
- IE（注：非 Edge，数据参考自 IE 9 至 IE 11）
- 火狐
- Safari

无头浏览器（Headless Browsers）

无头浏览器是一种没有用户图形界面的 WEB 浏览器，我们可以用命令行界面编程控制该浏览器，达到自动化运行 Web 页面（比如功能测试、网站检索、单元测试等）的目的。可将无头浏览器看作网页浏览器，不过你用命令行来检索、遍历网页。

最常见的无头浏览器：

- PhantomJS
- slimerjs
- trifleJS

Webviews

原生操作系统的原生应用中，用 Webviews 来运行网页。不妨把 webview 想成一个嵌进原生应用的 iframe 或 WEB 浏览器标签，而该原生应用运行在设备的系统上（比如 IOS、安卓、windows）。

最常见的 webview 开发解决方案如下：

- Cordova (通常用于手机、平板的原生应用)
- NW.js (通常用于桌面应用)
- Electron (通常用于桌面应用)

基于 WEB 技术的原生应用

最终，前端开发者从 WEB 浏览器开发中得到经验，并可以脱离浏览器引擎环境编码。近来，人们正在构思如何脱离 web 引擎，用 web 技术（比如 CSS 和 JS）来构建原生应用。

该环境的例子：

- NativeScript
- React Native

注：

请确认自己明白 “web platform” 的准确含义。查阅 “The Web platform: what it is” 和维基百科 “Open Web Platform”

2016 前端开发回顾

- UI 组件和组件树被用于构建复杂 UI。
- 组件由单一文件构成，在单个文件中可能同时包含 HTML、CSS 和 JS 不再有违主流开发思想。
- React、Redux、Webpack、ECMAScript 2015（也叫 ES6）和 Babel 被广泛采用。这些解决方案跃居于最常用技术榜单前列。
- 开发者意识到，在开发原生应用时，借助 webviews 的 H5 混合式移动开发在多数情况下不具备足够优势。
- React Native 和 NativeScript 开始替代 H5 混合式 webview 开发。
- 大多数人舍弃 Gulp 转而使用 NPM 脚本，但 Gulp 仍受欢迎。
- SASS 工具继续受到欢迎，与此同时 PostCSS（+ CSSNext）开始发展。
- 大多数开发者都开始对 HTML、CSS 和 JavaScript（ESlint 替代了 JShint，JSCS 也被整合进 ESLint）进行语法检查。
- 开发者弃 Sublime 和 Atom 转投 Visual Studio Code 编辑器，这成为一种趋势。
- jQuery 仍有热度，但使用率和关注度都在下滑。jQuery 3 已然发布 ，却无人问津。
- Vue.js 理所应当地吸收更多追随者。
- JavaScript 函数式编程和模式备受关注。
- 离线开发和渐进式 WEB 应用（PWA）步入主流。
- 微软发力。
- 基于 web 技术，使用 NW.js 和 Electron 开发 windows，OSX 和 linux 原生应用的方式逐渐成型。
- Angular 2 （在将来也叫作"Angular"）跌下神坛，多数人意识到它将不再如 Angular 1 那般辉煌了。
- JavaScript 大体上保持软件技术的中流砥柱位置。
- 更多的开发者开始把工具化（比如自动化）和测试当回事了。
- 静态站点生成器被重视起来。
- CSS 网格布局（CSS Grid）势头正旺且前途无量。
- NPM 受到来自 Yarn 的挑战。
- 下一代类 React 方案的演化通过 Preact、Deku、Rax 和 inferno 的形式展现，并伴随着少量 API 改动。
- 此前大多数人学习接受 JSX，而如今他们已经享受其中。
- 一种可用的 CSS 模块模式（CSS encapsulation）已经实现并投入使用，因此对许多人来说，CSS in JS 成为一种切实可行的解决方案。
- 越来越多人着手进行 UI 的功能性、整合性测试，其中包含例如可视化 CSS 和 RWD（译注：响应式网页设计，全称 Responsive web design）回归测试的概念。
- 得益于老版本 IE 使用、开发程度的大幅度降低，为浏览器 API 一致性而战的时代已离我们远去。

- 几乎人人都意识到开发网页的时候必须考虑多设备适配策略。
- 使用其他语言的开发者持续涌入 JS 领域，他们也带来了一些东西：例如类型检测，和对类语法以及面向对象思想的执念。
- 前端开发引入了热模块替换技术和时间旅行调试。
- 原生 JS 浏览器模块加载器更受期待了。
- Enforcing CSS 和 JS 格式规范变得更受重视（就 ES3 到 ES6 编码以及 CSS 预处理语法两者的变化而论）。
- 少部分开发者开始在 JS 上跑极限学习机（Extreme Learning Machine）算法，这足以引起注意。
- TypeScript 被正式使用在一些地方，并且圈了一些粉。
- aurelia 成为企业级开发者的明智之选（也就是说受到支持！）。
- Webpack 采取措施并巩固了优势地位，更胜一筹的 JSPM 解决方案暂居其下。
- HTTPS，嗯，这个我们很重视
- BASH 在 windows 系统上展露头脚。
- 通知功能 API 可以被使用了，并在 chrome 上有些滥用，但这只会发生在你授予它权限之后。
- FireBug 调试工具退出历史舞台。
- 2016年，CSS 20 岁了。
- Immutability 概念发展势头正旺。

展望2017

- Web Assembly 有望达到一个新的高度。
- 有望在 `<script> </script>` 中使用 `import` 进行模块懒加载。详情见这里
- JavaScript 同构解决方案持续增长，致敬服务器端输出前端内容的时代（即：页面直出到浏览器）。参见 NextJS
- 响应式编程继续茁壮成长。（参见 MobX and RxJS）
- React，尤其是它倡导的概念继续占有支配地位。而 React 本身会被彻底重写（React Fiber）或者进化（Inferno）
- Angular 终于决定遵循 SEMVER 规范，所以 Angular4（甚至于 Angular5）有望在 2017 年发布。 参见 Roadmap
- 简单的网站即 Web 1.0 可能会重新流行，但会建立在 2017 年新工具的基础上。（例如 Static site generation）
- RESTful JSON APIs 会更有竞争力（参见 GraphQL）
- 2017 可能是 VueJS 的丰收年。
- 越来越多的开发者在做静态站点以及 API CMS tools 时开始抛弃传统的 CMS 解决方案。
- 更多的人从 Sass 转向 PostCSS + CSSNext。
- 越来越多地见到 HTTP2 和 HTTPS 的身影。
- Web Components 继续潜伏等待开发者们助力实现前所未有的大爆发。
- 无框架的框架势头正猛。（参见 Svelte）
- JavaScript 标准即将尘埃落定，同时期待 CSS 也能迎来大爆发，并早日稳定下来， 否则开发者们始终惶惶不可终日。
- 相对于开放的 Web，对于 App Store 的仇恨与日俱增。
- Redux 继续接受来自竞争对手的激烈挑战。 （参见 Mobx）
- YARN 将赢得更多的粉丝。
- "Front-end apps"、"Thick Client apps"、"Static apps"、"No Backend app"、"SPA's"、"Front-end driven app" 这些理念可以归结为一个概念："JAM Stack"。

第一部分. 前端实践

第一部分概述了前端工程实践。

前端职称

以下是前端职称的描述清单。前端开发者最常用的称呼是“前端开发者”或“前端工程师”。注意，通常在名称里包含“前端”、“客户端”、“web UI”、“HTML”、“CSS”和“JavaScript”的工作意味着就职者在 HTML、CSS、DOM 和 JavaScript 方面有一定的技术深度。

前端开发者

这是一个通用的职称，它描述的是熟悉 HTML、CSS、DOM 和 Javascript 并在 web 平台加以实践的开发者。

前端工程师（又叫 **JavaScript** 开发者或全栈 **JavaScript** 开发者）

该职称授予有计算机科学、工程背景并能熟练运用相关技能的前端开发者。该职位通常要求就职者有计算机科学学士学位和若干年软件开发经验。当职称名中还包含“JS 应用”时，意味着就职者是高级 JS 开发者，他拥有高级编程、软件开发和应用开发这些技能（也就是有若干年构建前端应用的经验）

CSS/HTML 开发者

该前端职称描述的是熟练掌握 HTML 和 CSS 技术的开发者，但对 JavaScript 和应用技术不作要求。

WEB 前端设计师

当职称名包含“设计师”时，意味着该设计师拥有前端技能（也就是 HTML 和 CSS）及专业设计（视觉设计和交互设计）技能。

Web/前端用户体验（又称 **UI**）开发者**/**工程师

当职称名包含“交互”或“UI”时，意味着该开发者除了拥有前端开发或前端工程能力以外，还拥有交互设计能力。

手机/平板前端开发者

当职称名包含“手机”或“平板”时，意味着该开发者拥有在手机或者平板设备（可以是原生，或者 web 平台，也就是在浏览器里）上进行前端开发的经验。

前端 SEO 专家

当职称名包含“SEO”时，意味着该开发者熟悉用前端技术设计 SEO 策略（搜索引擎优化策略）。

前端访问性专家

当职称名包含“访问性”时，意味着该开发者熟悉使用前端技术处理访问性的要求和标准。

前端运维

当职称名包含“运维”时，意味着开发者在涉及合作、整合、部署、自动化和测试的软件开发实践上有丰富的经验。

前端测试／质量保证（QA）

当职称名包含“测试”或“质量保证”时，意味着该开发者熟悉测试和管理软件（涉及到单元测试、功能测试、用户测试和 A/B 测试）。

注意，如果职称名中有“全栈”或“Web 开发”，招聘者可能是用这些词来描述负责整体 web/app 开发的职位，也就是同时负责前端（可能包含设计）和后端的职位。

前端开发者所使用的技术



图片来源: *http://www.2n2media.com/compare-front-end-development-and-back-end-development*

前端开发者所使用的核心技术有如下这些（建议按顺序进行学习）：

1. 统一资源定位符 (URLs)
2. 超文本传输协议 (HTTP)
3. 超文本标记语言 (HTML)
4. 层叠样式表 (CSS)
5. JavaScript 编程语言 (ECMAScript 262)
6. JavaScript 对象表示法 (JSON)
7. 文档对象模型(DOM)
8. 网络 APIs (HTML5 或者浏览器 APIs)
9. 网络内容可达性指南 (WCAG) & 可访问的富互联网应用 (ARIA)

下面的介绍涵盖了以上技术的定义，相关文档以及具体规范。至于更加详尽的网络开发规范请参看 platform.html5.org。

超文本标记语言 (HTML)

> 超文本标记语言，通常被称为 HTML，被用作创建网页的标准标记语言。网络浏览器可以读取 HTML 文件并且把它们渲染成可见或可听的网页。HTML 在语义上描述了一个网站的结构，并且隐含了其表现形式，因此是一种标记语言，而非程序语言。
>
> — 维基百科

相关规范／文档：

- W3C HTML 规范大全
- 动态标准中的 HTML 元素
- 全局属性
- W3C 的 HTML 5.2
- HTML 属性参考资料
- HTML 元素参考资料
- HTML 语法 from the Living Standard

层叠样式表 (CSS)

> 层叠样式表（CSS）是一种样式语言，用来描述使用标记语言编写的文档的外观和格式。尽管样式表通常被用来改变以 HTML 和 XHTML 的方式编写的网页和用户界面的样式，它也能被运用在任何使用 XML 编写的文档中，其中包括 XML，SVG 和 XUL。同 HTML 和 Javascript 一样，CSS 是用以构建具有视觉冲击力的网页和用户界面的基础技术。
>
> — 维基百科

相关规范／文档：

- W3C CSS 规范大全
- 层叠样式表 2.2 (CSS 2.2) 规范
- CSS 参考资料
- 第三代选择器

文档对象模型 (DOM)

> 文档对象模型（DOM）是一个跨平台并且具有语言无关性的概念，用来表示 HTML，XTHML 以及 XML 文档中的对象以及这类对象的交互方式。每份文档中按照树形结构进行组织的节点，被称为 DOM 树。可以使用 DOM 树中的对象所拥有的方法对该对象进行处理和操作。DOM 的 API 规定了它的公共接口。
>
> — 维基百科

相关规范／文档：

- 文档对象模型 (DOM) 的第三代事件规范
- DOM 动态标准
- W3C DOM4

JavaScript 编程语言 (ECMAScript 262)

> JavaScript 是一个高级的、动态的、弱类型的解释性编程语言，被包含在 ECMAScript 的语言规范中。同 HTML 和 CSS 一样，它是万维网内容生产环节必不可少的三种技术之一，被大多数网站所使用，并且在不需要使用插件的情况下被所有现代的浏览器所支持。Javascript 基于原型并且把函数视为头等公民，因此是一种多范式的编程语言，支持面向对象，命令式以及函数式编程风格。它有一个可被用来操作文字，数组，日期以及正则表达式的 API，然而并不包含任何 I/O，因此像建网，存储或者图形工具之类的功能就需要依赖它所在的开发环境。
>
> — 维基百科

相关规范／文档：

- ECMAScript® 2017 语言规范

Web APIs (HTML5 及其他)

> 当使用 Javascript 给网页编写代码的时候，有许多的 API 可供使用。以下是一张关于所有在开发网站或者网络应用时可以使用的公共接口的表。
>
> — *Mozilla*

相关文档：

- Web API 公共接口

超文本传输协议 (HTTP)

> 超文本传输协议（HTTP）是一个为分布式的、协作的、多媒体的信息系统指定的应用协议。HTTP 是万维网数据交流的基础。
>
> — 维基百科

相关规范：

- 超文本传输协议 -- HTTP/1.1
- HTTP/2

统一资源定位符 (URL)

> 统一资源定位符（URL）也被称为网址，是关于资源的引用，明确了计算机网络资源的地址以及检索该资源的机制。URL 是统一资源标志符（URI）的一种特定类型，尽管许多人认为这两个概念可以互换。URL 表明了获取所需资源的方式，然而并非每个 URI 都会如此。URLs 通常被用于网页（http），与此同时也被用作文件传输（ftp），邮件（mailto），数据库接入（JDBC），以及许多其他的应用。
>
> — 维基百科

相关规范：

- 统一资源定位符 (URL)
- URL 动态标准

JavaScript 对象表示法 (JSON)

> Javascript 对象表示法是在异步的浏览器／服务器交流方式（AJAJ）中所使用的主流的数据格式，并且基本上已经取代 XML（被 AJAX 所使用）。虽然 JSON 是从 Javascript 中衍生出的数据格式，但它实际上是独立于语言的。在许多的编程语言中都有现成的，用于解析和生成 JSON 数据的代码。JSON 数据格式的规范最开始是由 Douglas Crockford 制定的，现在被两个互相竞争的标准所描述：RFC 7159 以及 ECMA-404。ECMA 标准更轻量，仅仅规定了可以使用的语法规则；而 RFC 则基于句法上的以及安全上的考量提供了更细致的规范。JSON 官方的网络媒体类型是 application/json。JSON 文件名的后缀是 .json。
>
> — 维基百科

相关规范：

- 介绍 JSON
- JSON API
- 数据交换格式 JSON

网络内容无障碍指南 (WCAG) & 可无障碍访问的富互联网应用 (ARIA)

> 可达性讨论的是对残障人士友好的产品设计，设备，服务或者环境。无障碍设计通过使用对残障人士友好的技术（例如计算机屏幕读取仪）对“直接访问”（例如，普通人的访问）和“间接访问”进行兼容。
>
> — 维基百科

- 可无障碍访问的富互联网应用 (WAI-ARIA) 现状
- 网络无障碍倡议 (WAI)
- 网络内容无障碍指南 (WCAG) 的现状

前端开发技能



图片来源：*http://blog.naustud.io/2015/06/baseline-for-modern-front-end-developers.html*

假设每个领域的前端开发者都使用基础的高级 HTML、CSS、DOM、JavaScript、HTTP/URL 和浏览器技能。

除了 HTML、CSS、DOM、JavaScript、HTTP/URL以及浏览器开发的专业知识，一个前端开发者还需要掌握以下一项或多项技能：

- 内容管理系统（亦称 CMS）
- Node.js
- 跨浏览器测试
- 跨平台测试
- 单元测试
- 跨设备测试
- Accessibility / WAI-ARIA
- 搜索引擎优化（亦称 SEO）
- 交互或用户界面设计
- 用户体验
- 适用性
- 电子商务系统
- 门户系统
- 线框绘制
- CSS 布局/ Grids
- DOM 操作（比如 jQuery）

- 移动 Web 性能
- 负载测试
- 性能测试
- 渐进增强/优雅降级
- 版本控制（比如 GIT）
- MVC / MVVM / MV*
- 函数式编程
- 数据格式（比如 JSON，XML）
- 数据API（比如 Restful API）
- Web 字体嵌入
- 可缩放矢量图形（亦称 SVG）
- 正则表达式
- 内容策略
- Microdata / Microformats
- 任务管理器，构建工具，过程自动化工具
- 自适应网页设计
- 面向对象的程序设计
- 应用程序构建
- 模块
- 依赖管理
- 包管理
- JavaScript 动画
- CSS 动画
- 图表/图形
- UI 控件
- 代码质量测试
- 代码覆盖率测试
- 代码复杂性分析
- 集成测试
- 命令行/命令行界面
- 模板策略
- 模板引擎
- 单页应用
- XHR 请求（亦称 AJAX）
- Web /浏览器 安全
- HTML 语义
- 浏览器开发工具

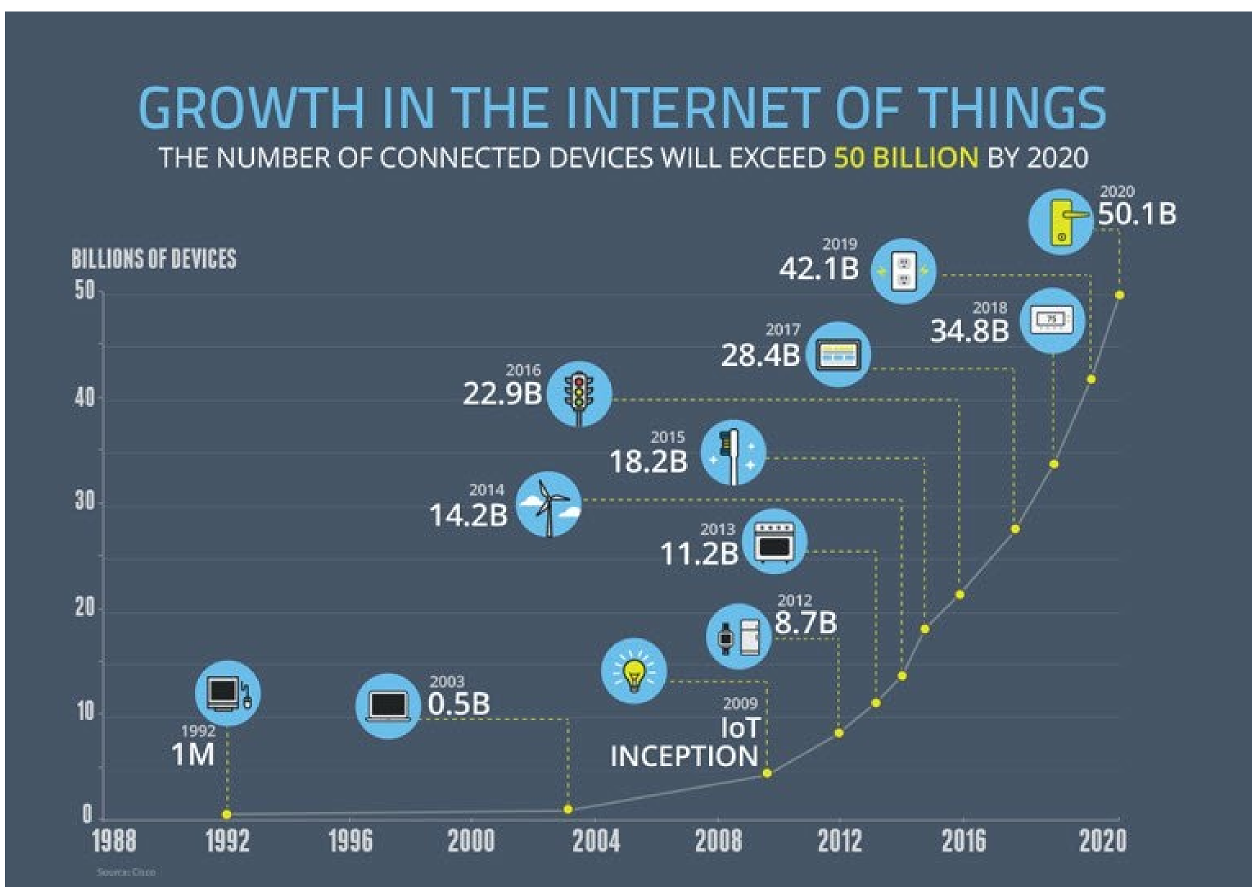
前端开发者为 ... 而开发

前端开发者所编写的 HTML、CSS 和 JS 代码，通常运行在基于下列某种操作系统（或称 OS）的 web 平台(比如 web 浏览器)上运行：

- Android
- Chromium
- iOS
- OS X
- Ubuntu (or some flavor of Linux)
- Windows Phone
- Windows

这些操作系统通常运行在下列一种或者多种设备上：

- 台式机
- 笔记本／上网本
- 手机
- 平板
- 电视
- 手表
- 其他东西（即汽车、冰箱、灯具、温控器等任何你能想到的东西）



图片源: *https://www.enterpriseirregulars.com/104084/roundup-internet-things-forecasts-market-estimates-2015/*

总体来说，前端技术可以在上述操作系统上运行，也能在使用下列运行时 web 平台方案的设备上运行：

- web 浏览器（例如：Chrome, IE, Safari, 火狐）
- 无头浏览器（例如：phantomJS）
- 拥有原生 API 桥梁的运行环境，被嵌入原生应用的 WebView／浏览器标签（想想 iframe）。通常 WebView 应用包含用 web 技术（也就是 HTML、CSS 和 JS）构造的 UI。 (例如: Apache Cordova、NW.js、Electron)
- web 技术构建的原生应用，该类 web 技术被整合在含原生 API 桥梁的运行环境中。其 UI 使用原生部分（比如 IOS 原生控件）而非 web 技术(例如: NativeScript、React Native)。

团队里的前端

通常，前端开发者是团队中的唯一角色，他们设计并开发 web 站点、web 应用或基于 web 技术的原生应用。

为了构建专业网站或 web 平台软件应用，一个基本的开发团队通常至少包含以下岗位。

- 视觉设计师（也就是字体、颜色、间距、情调、视觉概念和主题）
- UI／交互设计师／信息架构师（也就是线框、所有用户交互和 UI 功能、的指定，信息的架构）
- 前端开发者（也就是编写运行在客户端／设备上的代码）
- 后端开发者（也就是编写运行在服务端的代码）

这些岗位参照技能重叠来排序。通常前端开发者和后端开发者一样能较好地处理 UI／交互设计。团队成员通过接手重叠部分的职责，来担任多于单人的职责，这并不是稀罕的事。

假设上述团队是由项目经理或一些项目委托人（也就是参与人、项目管理者、项目经理等等）来指挥。

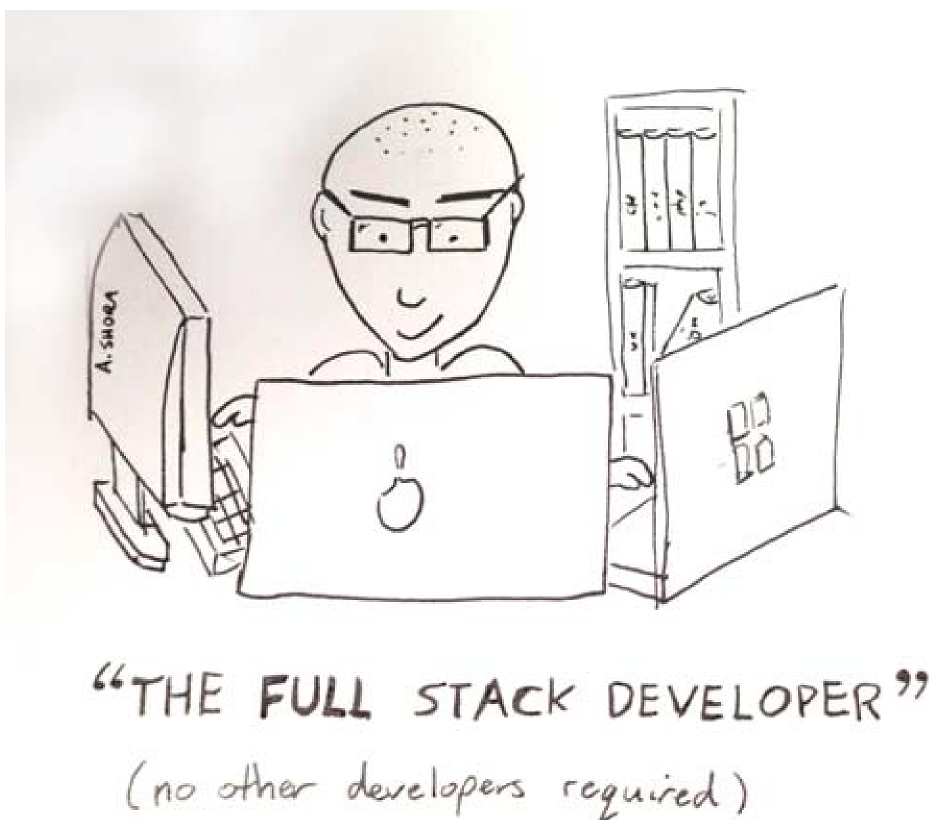
大型 web 团队可能包含下列岗位，这些岗位尚未提及：

- SEO 策略师
- 运维工程师
- API 开发者
- 数据库管理
- QA 工程师／测试者

注意:

一个小趋势似乎正在发生："全栈开发者" 同时担任前端和后端开发者的职责。

全能型人才/全栈神话



图片来源：*http://andyshora.com/full-stack-developers.html*

一个能够设计和开发整个 web 解决方案的人，需要具备非常强的专业技能和在视觉设计、UI/交互设计、前端开发、后端开发等领域非常丰富的经验。能这四个领域里精通一门或多门技术的人，就已经可以说是非常稀有的人才了。

老实说，你应当努力成为，或者尽力聘请到这些领域之一的一个专家（例如：视觉设计、交互设计/信息架构、前端开发、后端开发）的专家。声称自己在上述领域是专家的人非常稀少，甚至少的出奇。

不过，考虑到 JavaScript 已经渗透到了整套技术栈的所有层面（例如：React, node.js, express, couchDB, gulp.js 等），找到一个会前端开发和后端开发的 JavaScript 开发者并没有那么难。通常，这些全栈开发人员只需要关心 JavaScript —— 不像先前那样荒谬（还要关心视觉设计、交互设计和 CSS）。虽说在我看来依然少的出奇，但至少不像以前那么难找。因此，我并不建议开发者开始转向全栈工程师。在少数情况下可能有好处，但就职业发展来说，我认为前端开发工程师还是应当着重关注前端相关的技术。

备注:

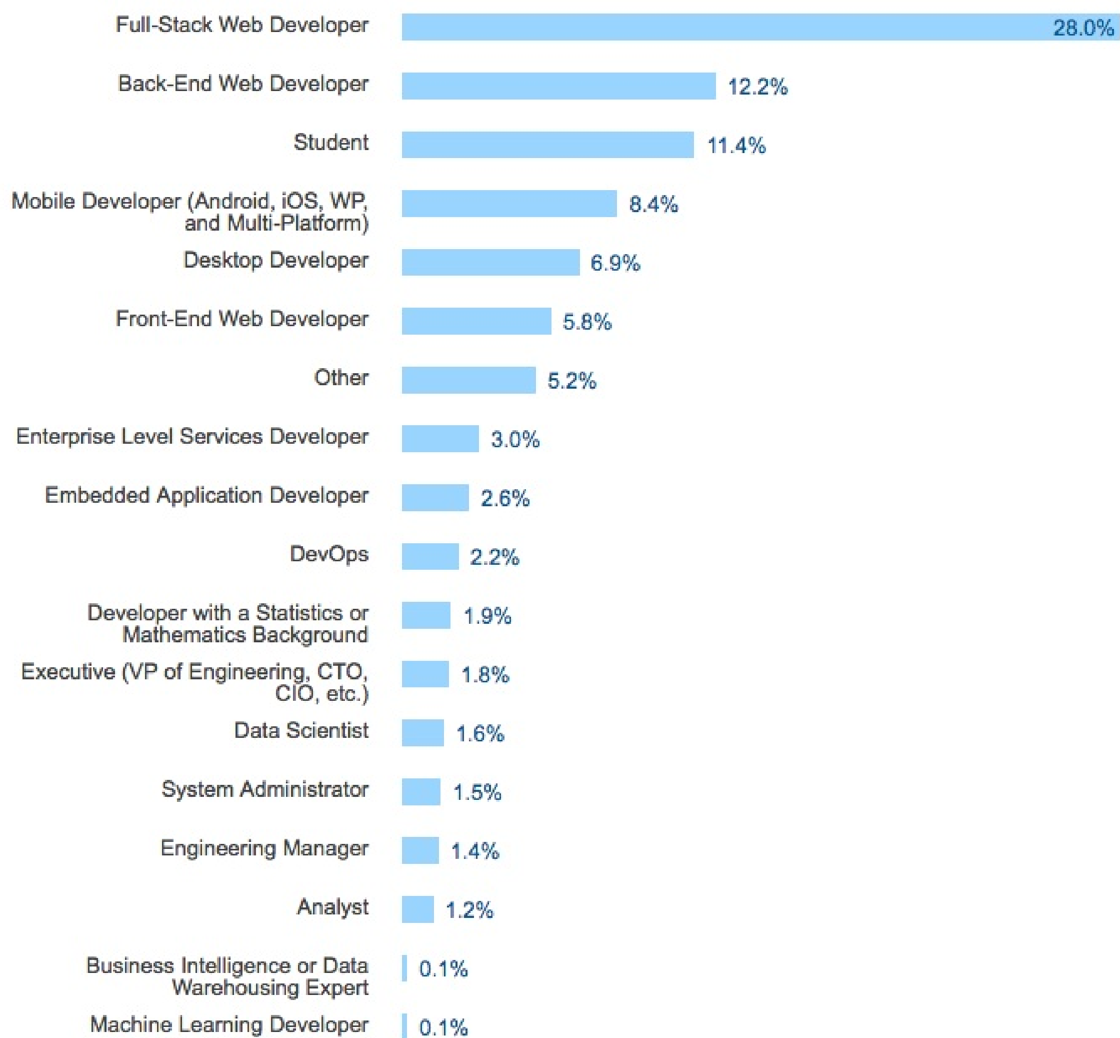
“全栈开发者”已经变为一个拥有多种含义的术语。也就是说使用这个术语时，它可能包含不止一层含义。分析一下下面的两个调查，从调查结果可以看出大多数开发者都是全栈开发者。但以我近 20 年的经验来看，这并不是真的。

Which one of these roles are you most interested in?

6696 out of 15624 people answered this question

1	Full-Stack Web Developer	2,570 / **38%**
2	Front-End Web Developer	1,376 / **21%**
3	Back-End Web Developer	704 / **11%**
4	Data Scientist / Data Engineer	644 / **10%**
5	Mobile Developer	414 / **6%**
6	User Experience Designer	275 / **4%**
7	DevOps / SysAdmin	219 / **3%**
8	Other	199 / **3%**
9	Product Manager	191 / **3%**
10	Quality Assurance Engineer	104 / **2%**

图片来源：*https://medium.freecodecamp.com/we-asked-15-000-people-who-they-are-and-how-theyre-learning-to-code-4104e29b2781#.ngcpn8nlz*



图片来源：*http://stackoverflow.com/research/developer-survey-2016#developer-profile-developer-occupations*

前端面试

Questions you may get asked:

- 每个 JavaScript 开发者都应当知道的十个问题
- 前端开发面试问题
- Web 前端开发小测验
- 为前端开发人员准备的面试问题
- JavaScript 在线小测验

Questions you ask:

- 开发者向未来老板提问的开源问题清单

Preparing:

- 2017 年 Web 前端开发面试前的准备
- Interview Cake [$]
- Cracking the front-end interview

前端开发职位公示

目前已经有大量的技术工作岗位存在，下面是目前与前端开发最具相关性的职位公示网站列表：

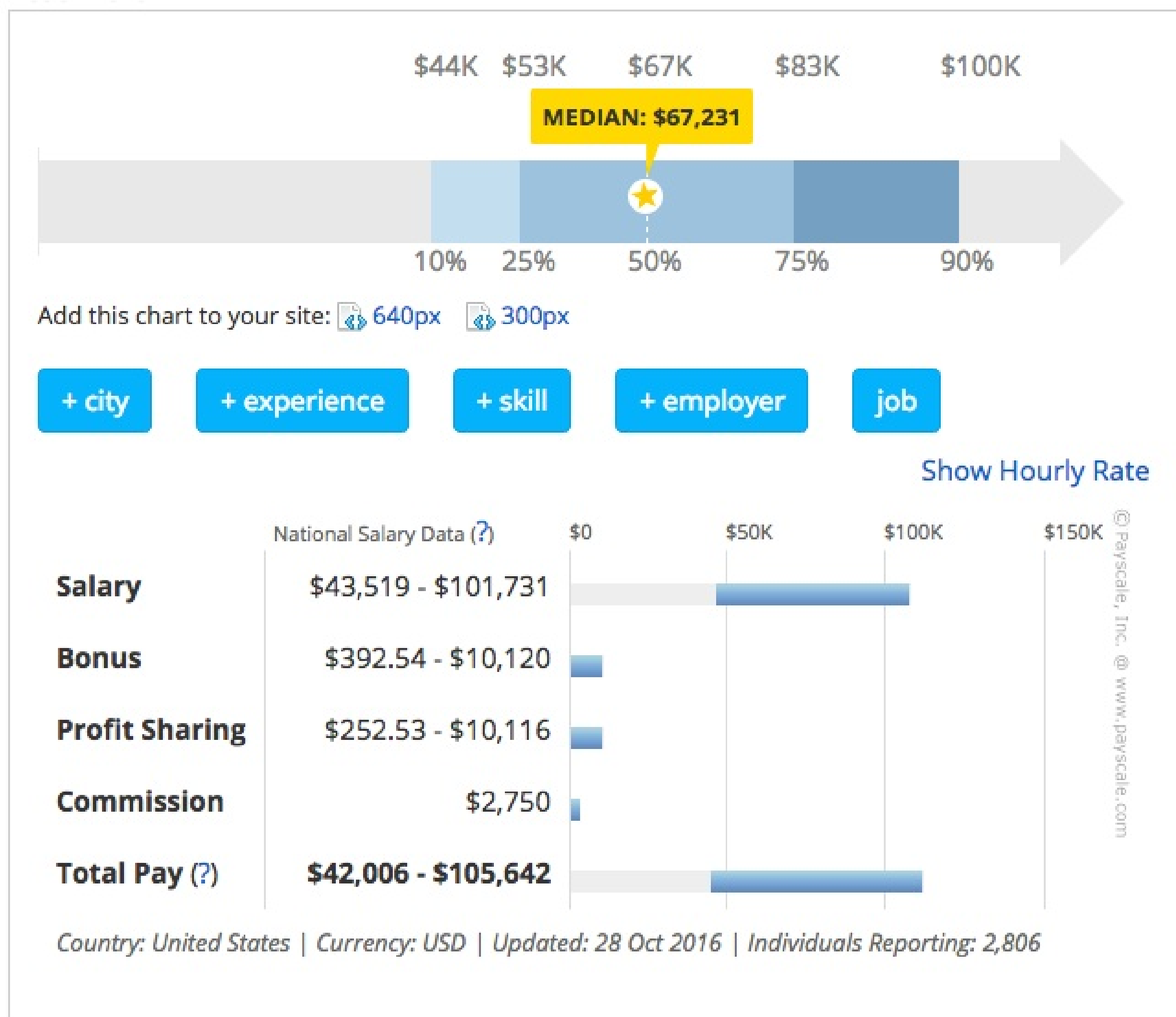
- angularjobs.com
- authenticjobs.com
- careers.stackoverflow.com
- css-tricks.com/jobs
- codepen.io/jobs/
- frontenddeveloperjob.com
- glassdoor.com
- jobs.emberjs.com
- jobs.github.com
- weworkremotely.com
- fronthat.com

NOTES:

如果你想找一个远程办公的前端工作，请移步 Remote-friendly companies

前端开发薪资

在美国，中等水平的前端开发者的平均年薪大约为 75K 美元。当然，在刚开始进入前端开发领域时，根据开发者的所在地区和工作经验，期望年薪应该在 35K 美元左右。

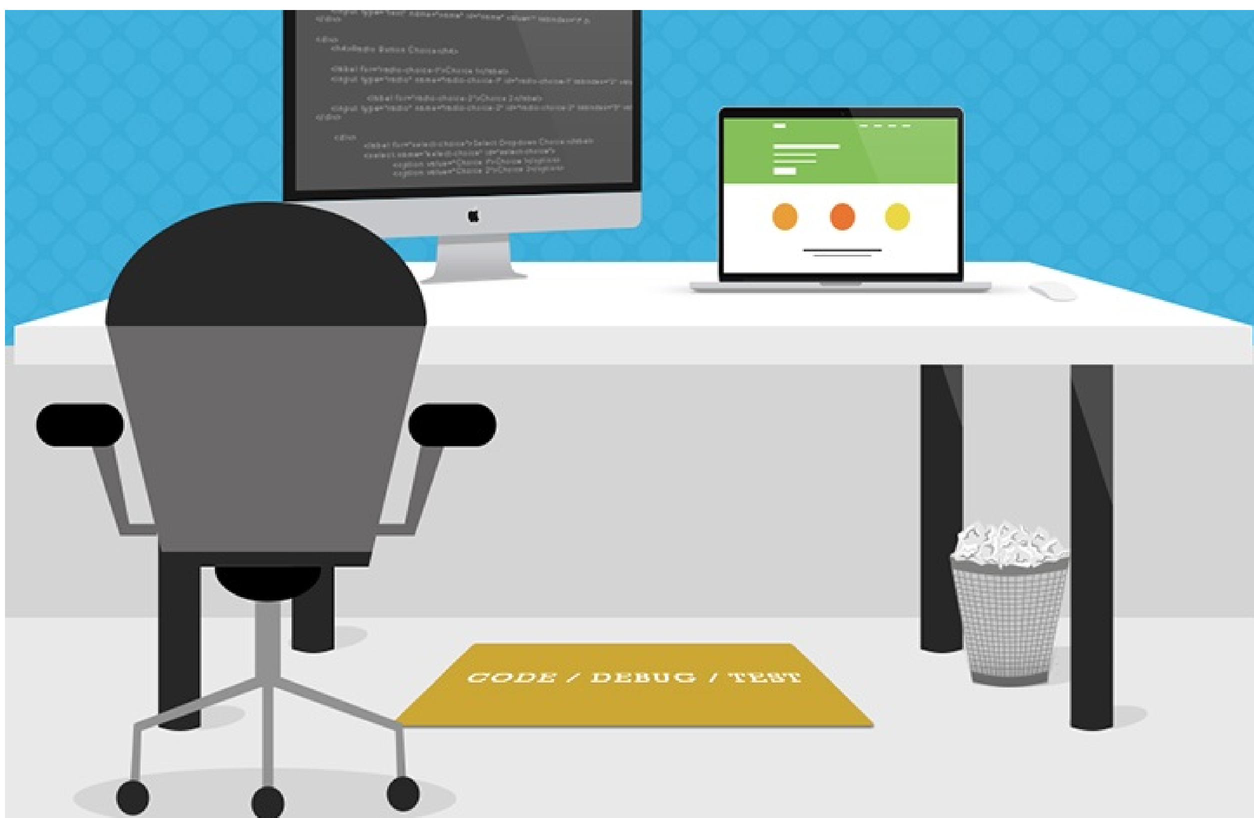


图片来源：*http://intersog.com/blog/chicago-tech-salary-guide-2015/*

注意：

高级前端开发者或工程师通常可以自愿选择办公地点（例如在家远程办公）并且年薪超过 150K 美元（登陆 angel.co 或在 Stack Overflow Jobs 上查看年薪 150K+ 美元的前端开发工作）。

前端开发者是如何诞生的



图片来源: *http://cdn.skilledup.com/wp-content/uploads/2014/11/life-of-front-end-developer-infographic-Secondary.jpg*

到底如何才能成长为前端开发者？这很难说。现如今，你仍然不能指望在大学毕业的时候拿到一个关于前端工程的学位。此外，我几乎没听说或见过一个前端开发者需要遭受一种过时的计算机科学学位或图形设计学位的折磨，才能专业地完成 HTML、CSS 和 JavaScript 编码工作。在我看来，如今大部分在前端圈工作的人，通常像是自学成才或来自非认证机构、课程及训练营。

如果你打算成为一个前端开发者，我将尽量让你跟随以下概括好的步骤（第二部分，"学习前端开发"，深入了解更多学习资源的细节）。

1. 请大致了解一下 WEB 的工作方式，并确认自己了解以下概念是什么，会在哪里出现：域名、DNS、URL、HTTP、网络、浏览器、服务器／托管、JSON、数据 API、HTML、CSS、DOM 和 JavaScript。你可以不求甚解，只需要大致理解各部分是如何协作的。请关注前端架构的主目录（high level outlines）。从编写简单的 web 页面和简要地学习前端应用（又叫 SPA）开始。
2. 学习 HTML。
3. 学习 CSS。
4. 学习 JavaScript。

5. 学习 DOM。
6. 学习 JSON 和数据 API。
7. 学习用户交互设计（也就是 UI 模式、UI 设计、用户习惯设计和可用性）的基础。
8. 学习命令行界面／命令行。
9. 学习软件工程的实践（也就是应用设计／架构、模版、Git、测试、监控、自动化、代码质量、开发方法论）。
10. 使用你觉得有用的东西（也就是 Webpack、React 和 Redux）配置和定制自己的工具箱。
11. 学习 Node.js。

对学习的简短建议：在学习工具和框架前应当先学习更底层的实现技术，学 DOM 而非 jQuery。学 CSS 而非 SASS。学 HTML 而非 HAML。学 JavaScript 而非 CoffeeScript。学 JavaScript ES6 模版而非 Handlebars。学 UI 模式而非使用 Bootstrap。

刚起步的时候，你应当对那些把 WEB 开发的复杂性掩盖掉的工具和框架保持警惕。高级工具和框架的不当运用会给人一种使用了高级技能的表象，而掩盖了这样的事实：开发者对于基础和底层概念所知寥寥。

本书的剩余部分会为你指出可能的资源，供你学习前端开发和实践开发所需的工具。在此过程中，我们的前提假设是你不仅学习新知识，而且学以致用并研究工具的使用方法。有人说实践出真知，还有人说学以致用，而我的建议是，请结合以上两个观点，找到合适自己的工作方式并践行它。但毫无疑问的是，要结合起来！因此别止步于看，要实践起来！学习，实践，学习，实践，学习，实践。要不断重复两者，以应对飞速变化的东西。这就是要学习基础知识而非高级工具的原因，学习基础知识十分重要。

近来出现了很多非认证的，收费昂贵的前端编码学院和训练营。这些成为前端开发者的途径，往往是由老师主导，由官方讲师制作的课程，它们遵循更传统的学习方式（也就是教学大纲、测试、小测验、项目、团队项目、分数等等）。如果你正在考虑是否要报一个收费昂贵的课程大纲，请牢记你是在互联网上！你可以在网上免费获得所有需要学习的东西。然而，如果你需要一个老师，要求这个老师能教你如何获取并学习到真正免费的知识，而且还要对你的学习负责，那你可能就要考虑一下系统化的课程了。否则，你学习任何专业技术所需的网络资源，每月收费几百块的视频资源会员，以及使你对知识怀有强烈的渴望，这些实际上都不是免费的。

如果你打算现在开始，那么请通过以下自学资源选择性地学习：

- 2016／2017 必须知道的 WEB 开发技术 [观看]
- 前端编程新手指南 [阅读 & 观看][免费]
- 如何成为前端开发者 [观看][收费]
- 前端课程 [阅读]
- freeCodeCamp [互动]
- 听说你想成为前端工程师 [观看]
- 启动 WEB 前端开发生涯 [观看][收费]

- WEB 前端开发：入门 [观看][收费]
- HTML、CSS、JS 快速入门 [观看][收费]
- WEB 开发介绍 [观看][收费]
- WEB 前端开发基础 [观看][收费]
- 学习前端工程 [观看][收费]
- 2015 前端[JS]开发者的基准 [阅读]
- 学习 WEB 前端开发 [观看][收费]
- 精通前端开发 [观看][收费]
- WEB 前端开发者微学位 [观看][收费]

第二部分：学习

第二部分给出了成为一名前端开发者需要学习的资源，分自学（指随时按自己的节奏来学习）类和教学（指规定次数和日期的正式课堂学习）类两种。

需要注意的是，不要因为某个学习资源被列出或某个学习类目包含在文档中，就代表前端开发人员要将它们全部掌握，我不建议那么做，没有意义。我提供的只是在这个领域可能需要被掌握的一些东西，在其中挑选你自己的那部分专业技能即可。

自学

本节重点介绍适合前端开发人员自我指导学习和自我指导职业发展的免费及付费资源（视频课程、图书等）。

以下所提及的资料包括免费和付费资料，其中付费资料将使用 [$] 标注。

作者相信，任何有决心且愿意付出的人都能够通过自我学习成为一名前端开发者。你只需要一台连接到互联网的计算机，以及一些用来购买书籍和视频课程的经费。

下面是一些视频学习网站（以技术为重点），我通常也推荐从这些站点中寻找学习内容：

- codecademy.com
- codeschool.com
- egghead.io
- eventedmind.com
- Frontend Masters
- Freecodecamp
- Khan Academy
- laracasts.com
- lynda.com [注意甄别，质量参差不齐]
- mijingo.com
- pluralsight.com [注意甄别，质量参差不齐]
- Treehouse
- tutsplus.com
- Udacity [注意甄别，质量参差不齐]

关于互联网／WEB

> 互联网是使用互联网协议族（TCP／IP）连接全球上亿设备的计算机网络系统。是由从地方到全球范围内几百万个私人的、学术界的、企业的和政府的网络所构成，通过电子，无线和光纤网络技术等等一系列广泛的技术联系在一起。互联网承载着规模可观的信息源和服务，例如相互关系的超文本文件和万维网（WWW）应用，电子邮件，电话和点对点文件共享网络。
>
> — 维基百科

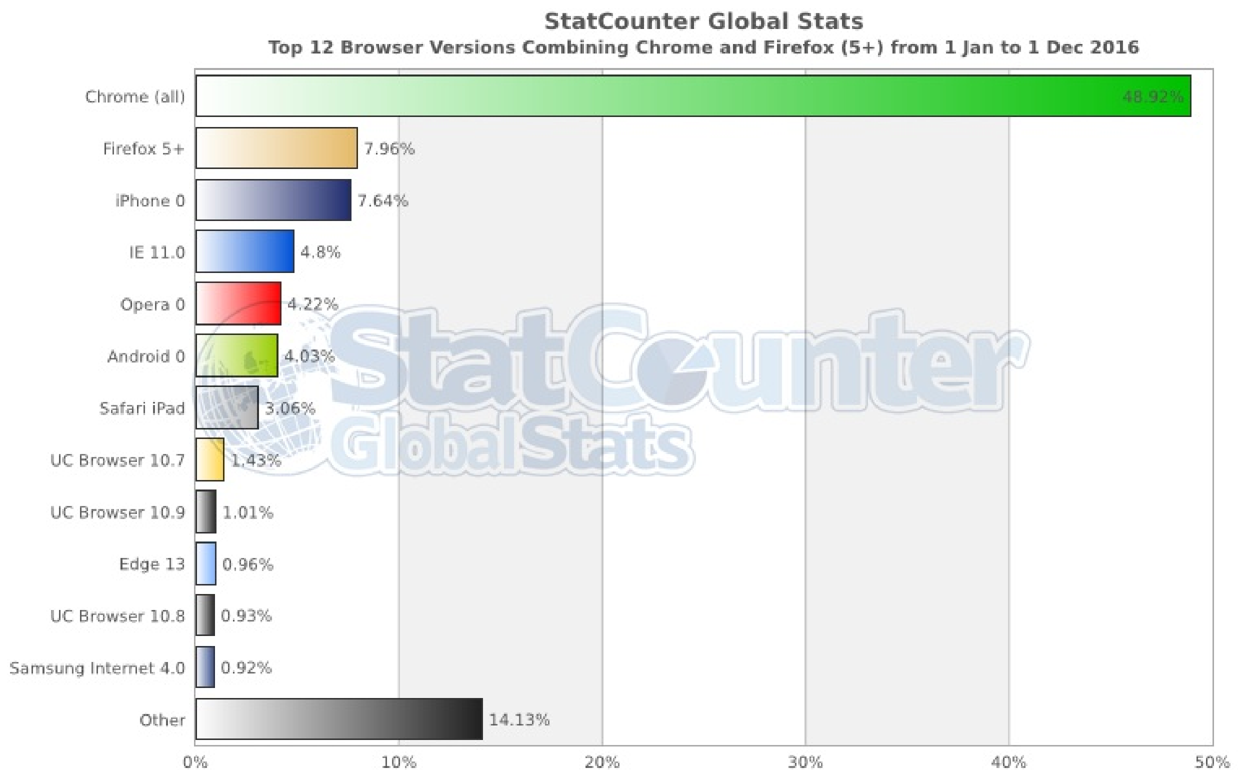
- 什么是互联网？ [视频]
- 互联网是如何工作的？ - W3C [文章]
- 互联网是如何工作的？ - Stanford Paper [文章]
- 互联网是如何工作的？ [视频]
- 互联网是如何在五分钟内工作的 [视频]
- 网络是如何工作的 [视频][付费]
- 什么是互联网？或“爱咋咋地，都一样” [文章]

学习 Web 浏览器

> Web浏览器 (通常被称为浏览器)是一个在万维网上检索、呈现、遍历信息资源的应用软件。信息资源是由统一资源标识符 (URI/URL) 来标识的，可以是网页、图片、视频或其他内容。资源中存在的超链接能够使用户轻松地将浏览器导航到相关资源。虽然浏览器主要目的是使用万维网，它们同样也可被用来获得专用网络中Web服务器所提供的信息，或者文件系统中的文件。
>
> — 维基百科

最常用的浏览器 **(任意设备上)** 如下：

1. Chrome (引擎: Blink + V8)
2. Firefox (引擎: Gecko + SpiderMonkey)
3. Internet Explorer (引擎: Trident + Chakra)
4. Safari (引擎: Webkit + SquirrelFish)



图片来源: *http://gs.statcounter.com/#all-browser_version_partially_combined-ww-monthly-201501-201601-bar*

浏览器和**Web**技术的演进（即**API**）：

- evolutionoftheweb.com [阅读]

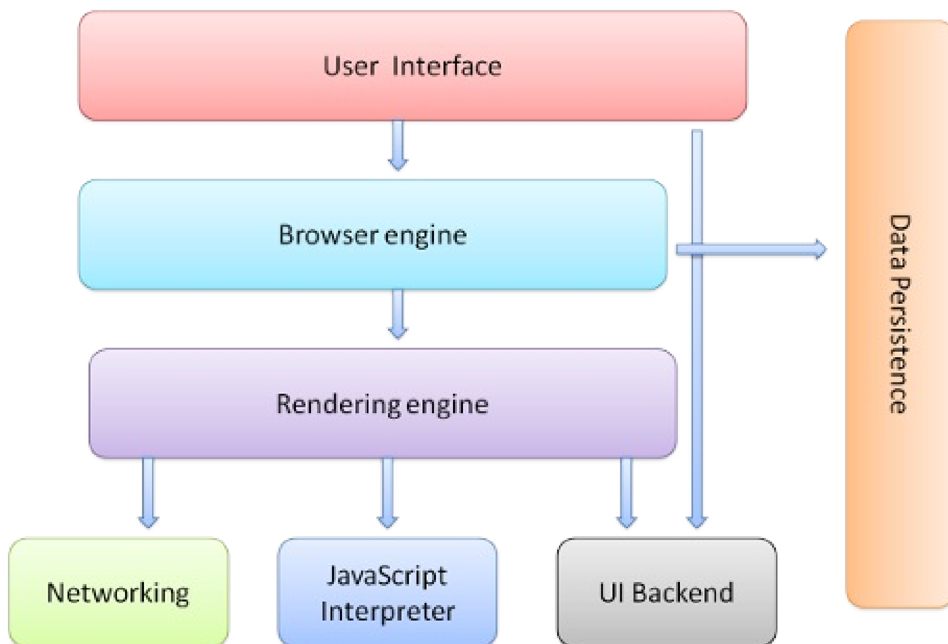
- Web浏览器的时间轴 [阅读]

最常用的**Headless(无GUI)**浏览器：

- PhantomJS (引擎: Webkit + SquirrelFish)
- SlimerJS (引擎: Gecko + SpiderMonkey)
- TrifleJS (引擎: Trident + Chakra)

浏览器是如何工作的：

- 了解关于浏览器和 Web 的20件事情 [阅读]
- 加速你的CSS：浏览器如何布局网页 [阅读]
- 浏览器如何工作：现代浏览器的幕后原理 [阅读]
- 浏览器是如何真正渲染一个网站的？ [观看]
- 什么会触发布局和重排 [阅读]
- 每位前端开发者必须知道的网页渲染技能 [阅读]



图片来源: *http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/*

浏览器优化

- 浏览器渲染优化 [观看]
- 网站性能优化 [观看]

对比浏览器

- Web浏览器的比较 [阅读]

浏览器 **Hacks**

- browserhacks.com [阅读]

浏览器开发

在过去，前端开发者要花费很多时间写代码来兼容几种不同浏览器。 与现在相比，这曾是一个巨大的问题。现在，第三方工具（例如：jQuery、React、 Post-CSS、Babel等等）结合现代浏览器使得浏览器开发相当容易。新的挑战不是用户使用哪个浏览器，而是他们选择哪种设备运行浏览器。

浏览器中的常青树

大多数现代浏览器的最新版本被认为是常青浏览器。也就是说，从理论上讲他们应该不需要用户操作自动更新。自动更新浏览器的这一举措已经反过来淘汰了不自动更新的旧版浏览器。

选择浏览器 [1]

如今，大多数前端开发人员使用Chrome和“Chrome”开发工具”开发前端代码。然而，最常用的现代浏览器都提供开发工具。使用哪个用于开发是一个主观的选择。最重要的问题是了解你必须支持哪个浏览器，哪个设备，然后适当地进行测试。

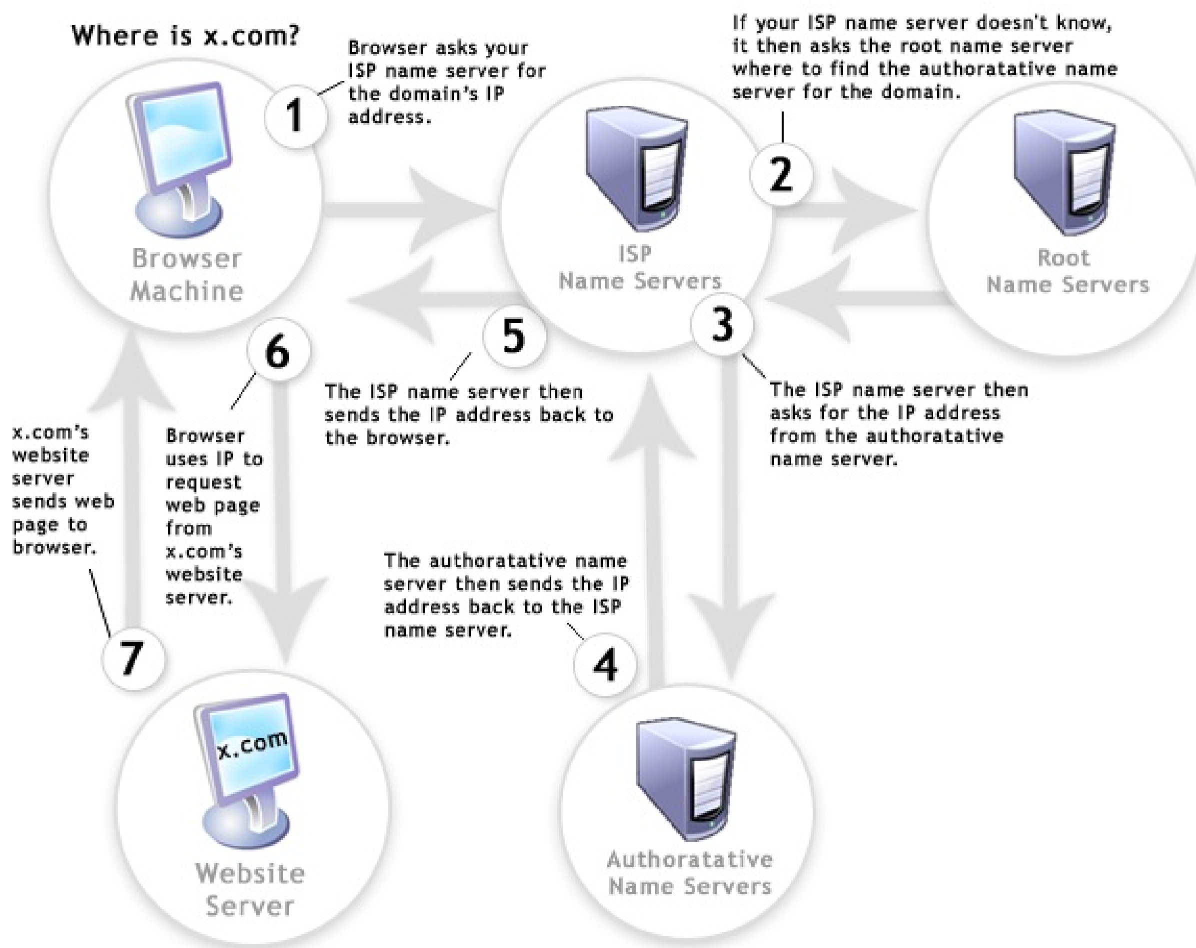
建议：

[1] 我建议使用Chrome，因为它的开发者工具在持续改进，并且包含了最强大的功能。

学习域名系统（又叫 DNS）

域名系统（DNS）是按层级划分的分布式命名系统，用以访问连接到互联网或私有网络中的计算机、服务、资源。它通过域名关联各种信息，这些域名被分配至每个参与实体（participating entities）上。它最突出的特点在于：将便于人们记忆的域名解析成数字表示的 IP 地址，IP 地址则符合全世界计算机服务器和设备的需求。网域名称系统是大多数互联网服务功能的重要组成部分，因为它是互联网主要的地址录服务。

> — *Wikipedia*



图片来源: *http://www.digital-digest.com/blog/DVDGuy/wp-content/uploads/2011/11/how_dns_works.jpg*

- 解释 DNS [观看]
- DNS 如何运行 [阅读]
- 互联网: IP 地址 和 DNS [观看]

学习 HTTP/Networks（包括 CORS 和 WebSockets）

> HTTP - 超文本传输协议（HTTP）是分布式、协作、超媒体信息系统的应用协议。HTTP是万维网数据通信的基础。
>
> — *Wikipedia*
>
> CORS - 跨域资源共享（CORS）是一种机制，允许网页向非本域下的资源提供者请求受限制的资源（比如字体）。
>
> — *Wikipedia*
>
> WebSockets - WebSocket是一种在单个 TCP 连接上进行全双工通讯的协议。WebSocket通信协议于2011年被IETF以 RFC 6455 的形式标准化了，Web IDL中的WebSocket API也被W3C定为标准。
>
> — *Wikipedia*

HTTP 规范

- HTTP/2
- 超文本传输协议 -- HTTP/1.1

HTTP

- 高性能浏览器网络：关于网络和Web性能开发者们应该的内容 [阅读]
- HTTP：最终指南（权威指南） [阅读][付费]
- HTTP/2 常见问答 [阅读]
- HTTP 基本原理 [观看][付费]
- HTTP/2 基本原理 [观看][付费]
- HTTP：Web开发者必须掌握的协议-第一部分 [阅读]
- HTTP：Web开发者必须了解的协议-第二部分 [阅读]
- HTTP 简介 [阅读]

HTTP 状态码

- HTTP 状态码
- 60秒学会HTTP状态码 [观看]

CORS 规范

- 跨域资源共享

CORS

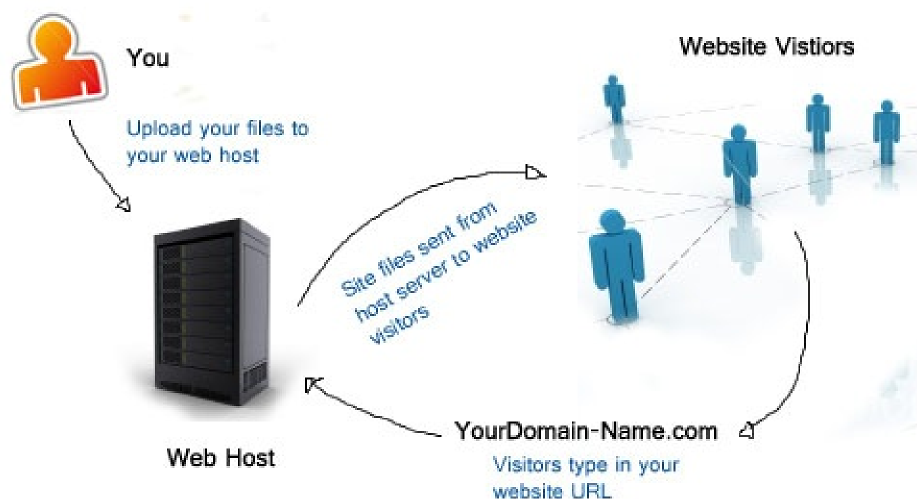
- CORS 实战 [阅读][付费]
- HTTP 访问控制（CORS） [阅读]

WebSockets

- 用websockets连接网络 [观看]
- WebSocket： 轻量级客户端-服务端通信 [阅读][付费]
- WebSocket协议 [阅读]

学习网页寄存（通称虚拟主机）

> 网页寄存服务是指一类 Internet 寄存服务，该服务允许组织和个人将他们的站点连接至互联网以供他人访问。网页寄存商在他们的数据中心提供可供用户购买或租用的主机，同时将这些主机接入互联网。网页寄存商也可以为数据中心中放置的其他公司服务器提供接入互联网的服务，这种服务称作主机托管（colocation），亦称 Housing in Latin America or France.
>
> — 维基百科



图片来源：*http://www.alphaelite.com.sg/sitev2/images/stories/webhostdemo.jpg*

入门学习：

- 网页寄存终极指南 [read]
- 网页寄存入门指南 [read]
- 网页寄存傻瓜教程 [read][$]

学习前端开发

常用资源:

- 前端[JS]开发者的基准线：2015 [阅读]
- 成为一名前端 Web 开发者 [观看][$]
- 成为一名 web 开发者的基本标准 [阅读]
- 前端 Web 开发基础 [观看][$]
- freeCodeCamp [互动]
- 前端课程 [阅读]
- 前端开发 精通 [观看][$]
- 前端 Web 开发 微学位 [观看][$]
- 前端 Web 开发 职业起步 [观看][$]
- 前端 Web 开发：开始 [观看][$]
- HTML5、CSS 和 JavaScript 的前端 Web 开发快速起步 [观看][$]
- 前端 Web 开发：Big Nerd Ranch 指引 [阅读][$]
- 前端指引 [阅读]
- Web 开发介绍 [观看][$]
- Isobar 前端编码标准 [阅读]
- 简洁前端工程 [观看][$]
- 学习前端 Web 开发 [观看][$]
- 设计前端 JS 应用 [观看]
- 听说你想当前端工程师 [观看]

常用前端时讯、新闻媒体 和 播客：

- Web 开发大舞台
- 前端开发技术周刊
- 前端开发 欢乐时光
- 前端开发 5分钟新闻
- 前端开发前沿
- 专注前端
- 前端时讯
- 移动 Web 技术周刊
- 开放式 Web 平台每日摘要
- Pony Foo 周刊
- 专家脱口秀
- Web 前沿
- Web 平台播客

- Web 工具每周一刊

学习用户界面／交互设计

> 用户界面设计 - 用户界面设计（UI），或者说用户界面工程，是指基于用户体验最优化，对机器以及软件，例如计算机，家庭电器，移动设备以及其他电子设备的用户界面所进行的设计。用户界面设计的目标是根据用户的使用目的（以用户为中心的设计），使交互尽可能简单而有效率。
>
> — 维基百科
>
> 交互设计模式 - 设计模式是对于常见设计问题解决方案的一种正式的总结。这是建筑师 Christopher Alexander 在进行城市规划和建筑设计的时候提出的概念，已经在很多其他的领域中被引用，例如教育，机构组建和软件设计。
>
> — 维基百科
>
> 用户体验设计 - 用户体验设计（UXD 或者 UED 或者 XD）是通过提升产品在同用户互动过程中的可用性，可达性和愉悦度，从而提升用户满意度的过程。用户体验设计包括传统的人机交互设计（HCI），但不仅限于此 —— 它涉及产品或服务的受众所能够感知到的方方面面。
>
> — 维基百科
>
> 人机交互 - 人机交互（HCI）研究计算机技术的设计和使用方法，尤其对人机交互起到桥梁作用的图形界面。HCI 方向的研究者既关心人类同计算机交互的方式，也关心能够使这种交互变得更为美观新颖的设计技术。
>
> — 维基百科

为了使大家有能力创建可用的用户界面，我会推荐以下的权威文档：

- 界面：交互设计原理 [阅读][增值付费]
- 为黑客而设计: 逆向工程之美 [阅读][增值付费]
- 献给非专业设计师的设计教程 [观看]
- 设计图形界面 [阅读][增值付费]
- 设计网络图形界面: 高级交互设计原理和模式 [阅读][增值付费]
- 别让我思考: 关于网络可用性的常识 [阅读][增值付费]

学习 HTML 和 CSS

> **HTML** —— 超文本标记语言，通常被称为 HTML, 是一种用于创建网页的标准标记语言。 网页浏览器可以读取HTML文件并将其渲染成可视化网页。HTML 描述了网页的结构语义，随着线索的呈现，使之成为标记语言而不是编程语言。
>
> — 维基百科
>
> **CSS** —— 层叠样式表 (CSS) 是一个层叠样式表语言，用来描述用标记语言来写的外观和格式。尽管经常都是用 HTML 和 XHTML 来改变网页和用户界面的样式，该语言可以应用于任何类型的 XML 文档，包括原生的 XML 、 SVG 和 XUL。 除了 HTML 和 JavaScript，CSS 是大多数网站的基础的技术，用来创建炫酷的网站，web 应用程序的用户界面和很多移动应用的用户界面。
>
> — 维基百科

就像建造房子一样，可以认为 HTML 用来搭建框架， CSS 用来绘画和装饰。

综合学习：

- CSS 里的绝对居中 [阅读]
- codecademy.com HTML 和 CSS [交互]
- CSS 定位 [视频][收费]
- 前端 web 开发：开始使用 [视频][收费]
- 前端Web开发 HTML5，CSS 和 JavaScript 快速入门 [视频][收费]
- HTML 和 CSS：设计和构建网站 [阅读][收费]
- HTML 文档流 [视频][收费]
- HTML 精通: 语义、规范和样式 [阅读][收费]
- 互联网很难 [阅读]
- HTML/CSS 简介：制作网站 [视频]
- 学习编写 HTML 和 CSS [阅读]
- 学习 CSS 布局 [阅读]
- MarkSheet [阅读]
- 语义 HTML：如何设计网页 [视频]
- 结实的 HTML 表单结构 [视频]
- 了解 CSS 盒模型 [视频]
- 弹性网页布局 [阅读]

掌握 **CSS**：

- Flexbox 完整指南 [阅读]

- CSS 烹饪 [交互]
- 从 CSS4 到 CSS1 的 CSS 选择器 [阅读]
- CSS 揭秘：更好的解决常见的网页布局问题 [阅读][收费]
- CSS3 [阅读]
- 深入 CSS3 [视频][收费]
- 什么是 Flexbox？！ 简单免费的20个视频教程帮助你掌握 CSS Flexbox [视频]

参考文献/说明:

- CSS 触发器... 布局、样式和混合使用的游戏, and Composite
- cssreference.io
- cssvalues.com
- Chrome 浏览器的默认 CSS
- Head - 可以在你的文档的 header 中的所有内容的列表
- HTML 属性参考
- MDN CSS 参考
- MDN HTML 元素参考

词汇表:

- CSS 词汇表 —— CSS 覆盖样式、属性和选择器的编程参考
- HTML 元素的 HTML 词汇表编程参考

标准/规范:

- 所有的 W3C CSS 规范
- 所有的 W3C HTML 规范
- CSS 2 级修订 2 （CSS 2.2）规范
- CSS 索引 —— 由 CSS 规范定义的属于列表
- 来自现存标准的 HTML 元素
- 全局属性
- HTML 语法 from the Living Standard
- 来自 W3C 的 HTML 5.2
- 3 级选择器

CSS 架构设计：

- 元素设计 [阅读]
- BEM
- ITCSS
- OOCSS [阅读]
- SMACSS [阅读][收费]
- 适用于CSS的可扩展模块化体系结构（ SMACSS ） [视频][收费]

- SUIT CSS
- rscss

设计/编写规范：

- CSS 编码指南 [阅读]
- css-architecture
- cssguidelin.es [阅读]
- Idiomatic CSS [阅读]
- MaintainableCSS [阅读]
- 开发有弹性的、耐用的和可持续的HTML 和 CSS 标准 [阅读]

HTML/CSS 简报：

- CSS 周报
- 聚焦前端

学习搜索引擎优化

> 搜索引擎优化(SEO)是一种针对搜索引擎非付费（又被称为自然检索、有机检索）或付费结果的处理过程，它会影响到网站及网页在这些结果中的可见度。通常来说，当一个网站更早的出现在搜索结果中（又或者在搜索结果中的排序更靠前），就会有更多的搜索引擎使用者对它进行访问。SEO 可以针对不同的搜索，包括图像搜索、本地搜索、视频搜索、学术搜索、新闻搜索和特定行业的垂直搜索引擎。
>
> — *Wikipedia*

综合学习：

- 谷歌搜索引擎优化入门指南 [阅读]
- SEO 基本原理 —— David Booth [视频][收费]
- SEO 基本原理 —— Paul Wilson [视频][收费]
- SEO 初学者教程 —— 2016 [阅读]
- 给网站设计师的 SEO 建议 [视频][收费]

学习 JavaScript

> JavaScript是一种高级的、动态的、无类型的、解释型的的编程语言。它是万维网内容制作的，除了HTML和CSS以外的三大基本技术之一；大多数网站都在使用它，并且所有现代的Web浏览器无插件的支持。JavaScript是基于原型并具备极好的功能，这使它成为一种多范式的语言，支持面向对象、指令式和函数式编程风格。它提供文字、数组、日期和正则表达式的操作API，但不支持任何I/O相关的，例如网络、存储或图形设备，这取决于它所嵌入的主机环境。
>
> — 维基百科

入门学习：

- codecademy.com JavaScript [互动]
- JavaScript 第一步 [阅读]
- 创建 JavaScript 代码块 [阅读]
- JavaScript Enlightenment [阅读]
- JavaScript 对象基础 [阅读]
- JavaScript 编程精解 [阅读]

一般学习：

- Speaking JavaScript [阅读]
- 你不知道的 JS: 入门 [阅读]
- 你不知道的 JS: 类型和语法 [阅读]
- 你不知道的 JS: 作用域和闭包 [阅读]
- JavaScript 中 'this' 的大致解释 [阅读]
- 你不知道的 JS: this 和 对象原型 [阅读]

配置：

- 架设 ES6 [阅读]
- 每个人都能学 ES6 [观看][$]
- Exploring ES6 [阅读]
- 你不知道的 JS: ES6及其上 [阅读]
- Understanding ECMAScript 6 [阅读]
- ES6 精粹 [观看][$]
- Exploring ES2016 and ES2017 [阅读]
- JS 正则表达式 [阅读]
- 使用正则表达式 [观看][$]
- 你不知道的 JS: 异步和性能 [阅读]

- JavaScript Promises [阅读][$]
- 测试驱动的 JavaScript 开发 [阅读][$]
- JS 流言 [阅读]

JavaScript 函数式编程:

- 函数式编程术语
- JavaScript 中的函数式编程 [观看]
- Functional-Light-JS [阅读]
- Functional Programming in JavaScript [阅读]
- JS 函数式编程指南 [阅读]
- JavaScript Allongé [阅读][$]
- JS函数式编程的核心 [观看][$]
- Functional-Lite JavaScript [观看][$]

参考文档：

- MDN JavaScript 参考
- MSDN JavaScrip 参考

术语表:

- JavaScript 辞书
- JavaScript 术语表
- JavaScript 简明辞书

标准／规范:

- ECMAScript® 2015 语言规范
- ECMAScript® 2016 语言规范
- ECMAScript® 2017 语言规范
- ECMA262的状态、进程和文档

样式:

- Airbnb JS样式指南
- JavaScript 标准样式
- JavaScript 半标准样式

JS报纸、新闻&简报:

- Echo JS
- ECMAScript Daily
- ES.next News

- FiveJS
- JavaScript Air
- JavaScript Jabber
- JavaScript Kicks
- JavaScript Live
- JavaScript Weekly
- JavaScript.com

弃用的 **JS** 学习资源:

- Crockford - 第一章：历史那些事 [观看]
- Crockford - 第二章：JS的产生 [观看]
- Crockford - 第三章: 函数 [观看]
- Crockford - 第四章：Ajax的变化 [观看]
- Crockford - 第五章：安全和性能 [观看]
- Crockford - 第六章：循环 [观看]
- JavaScript 模式 [阅读][$]
- JavaScript 面向对象精要 [阅读][$]
- JavaScript 模块 [阅读]
- 函数式编程：用Underscore.js来介绍函数式编程 [阅读][$]
- js和web精粹 [观看][$]
- 高性能JavaScript [阅读][$]
- 先进的JavaScript [观看][$]

学习 Web 动画

概论

- 高级 SVG 动画 [付费][视频]
- 探索 Web 动画 [付费][视频]
- 使用 Snap.svg 制作动效 [付费][视频]
- CSS3 和 HTML5 中的动画效果 [付费][视频]
- 用 CSS 创建动效 [阅读 & 视频]
- 现实生活中的 CSS 动画 [付费][视频]
- HTML5 + JavaScript 动效基础原理 [付费][阅读]
- 学习运用 JavaScript 制作动效 [阅读 & 视频]
- 动效现状 —— 2015 [视频]
- Javascript 动画制作：开发与设计 [付费][阅读]

标准规范:

- Web 动画效果

学习 DOM、BOM 和 jQuery

> **DOM** - 文档对象模型 (DOM) 是一个跨平台的、独立于语言的约定，能以对象的形式对 HTML、XHTML 和 XML 文档进行展示和交互。每一个文档的节点被放到一个树形结构中，这棵树被称为 DOM 树。
>
> — 维基百科
>
> **BOM** - 浏览器对象模型 (BOM)是一个浏览器规范约定，涉及到网络浏览器暴露出来的所有对象。不像文档对象模型，BOM 并没有实现的标准和严格的定义，所以浏览器厂商想怎么实现 BOM 都可以。
>
> — 维基百科
>
> **jQuery** - jQuery 是一个跨平台的 JavaScript 库，为了简化客户端 HTML 中的脚本编写而诞生。jQuery 如今是最流行使用的 JavaScript 库，排名前 1000 万的高访问量网站中，有 65% 都在使用它。
>
> — 维基百科

最理想的学习路径，但也是最难的，就是先学习 JavaScript，然后学 DOM，然后 jQuery。但还是要确保你做的事有意义。大部分前端开发者是在初次学习 jQuery 的时候，顺带着学习 JavaScript 和 DOM。但无论你选哪一条路，都不能将 JavaScript、DOM 和 jQuery 当做一个黑盒来看。

普通学习资源：

- Codecademy.com jQuery [观看]
- 文档对象模型 [阅读]
- HTML/JS: 打造可交互的页面 [观看]
- HTML/JS: 通过 jQuery 打造可交互页面 [观看]
- jQuery 启蒙 [阅读]

进阶：

- 高级 DOM 编程: 动态 Web 设计技术 [阅读][$]
- jQuery & 纯 DOM 脚本编程的高级 JS 基础 [观看][$]
- Douglas Crockford: 麻烦的 API - DOM 理论 [观看]
- DOM 启蒙 [阅读][$] 或者 在线免费阅读
- 修正 jQuery 常见 Bug [观看][$]
- 无 jQuery 的 JavaScript [观看][$]
- jQuery 小贴士和小技巧 [观看][$]

参考/文档:

- jQuery 文档
- 事件
- DOM 浏览器支持度
- DOM 事件浏览器支持度
- HTML 接口浏览器支持度
- MDN 文档对象模型 (DOM)
- MDN 浏览器对象模型
- MDN 文档对象模型
- MDN 事件一览表
- MSDN 文档对象模型 (DOM)

标准/规范:

- 文档对象模型 (DOM) Level 3 事件规范
- 文档对象模型 (DOM) 技术报告
- DOM 最新标准
- W3C DOM4

学习网页字体 & 图标

> 网页排版是指在万维网上字体的使用。在第一次产生 HTML 的时候，每个网页浏览器的设置都霸道地控制了字体和样式。直到 1995 年，网景引入了 `<font>` 标签，才有机制来控制单网页的字体显示，HTML 3.2 标准随即规范了这一行为。然而，使用者必须在电脑上安装标签声明的字体或有降级字体， 例如浏览器默认的 sans-serif 或单空格字型，会被替代使用。1996 年，第一个 CSS 规范发布，也提供了相同的功能。 1998 年，CSS2 规范发布了，并试图通过添加字体匹配、合成和下载来改进字体选择的过程。可是这些技巧并没有很大用处，于是在 CSS2中 移除了。然而，IE 在 1997 年发布的 4.0 版本中增加了对字体下载的支持。[1]后来，在 CSS3 的字体模块中包含了字体下载，从那时起，Safari 3.1，Opera 10 和 Mozilla Firefox 3.5 也实现了这个功能。这增加了网页排版和字体下载用法的趣味性。- 维基百科

综合学习：

- 漂亮的网页类型：来自谷歌网页字体的完美字体样式 [阅读]
- 学习网页字体 [阅读]
- 通过 @font-face 来认识网页字体 [阅读]
- 响应式排版 [观看][$]
- 网页排版 [观看][$]

学习可访问性相关知识

> 可访问性是指针对于残疾人相关的产品、设备、服务或环境的设计。可访问性设计的概念不仅能确保“直接访问”（也就是说，无需任何辅助），也提供了“间接访问”（的能力），这意味着可以与人类的辅助技术相融合（例如：电脑屏幕阅读器）。
>
> 可访问性可以被视为针对于某些系统或实体的“访问能力”。该概念的重点在于通过可访问性或使用可访问性使残疾人和有特殊需要的人实现接入和访问。然而，研究和开发可访问性是受益每一个人。
>
> 可访问性不能与可用性混淆，可用性是指定用户为实现特定的目标，在特定的使用环境中使用产品（服务或者环境）的程度。
>
> 可访问性和通用化设计有着非常密切的关系。在这里，通用化设计指的是能够在最大范围内和最大程度上，设计普通人和残疾人都能够使用的产品的过程。这种设计应该尽可能多地照顾到不同类型和不同程度的拥有身体缺陷的用户。这就是在创造无障碍环境。
>
> — 维基百科

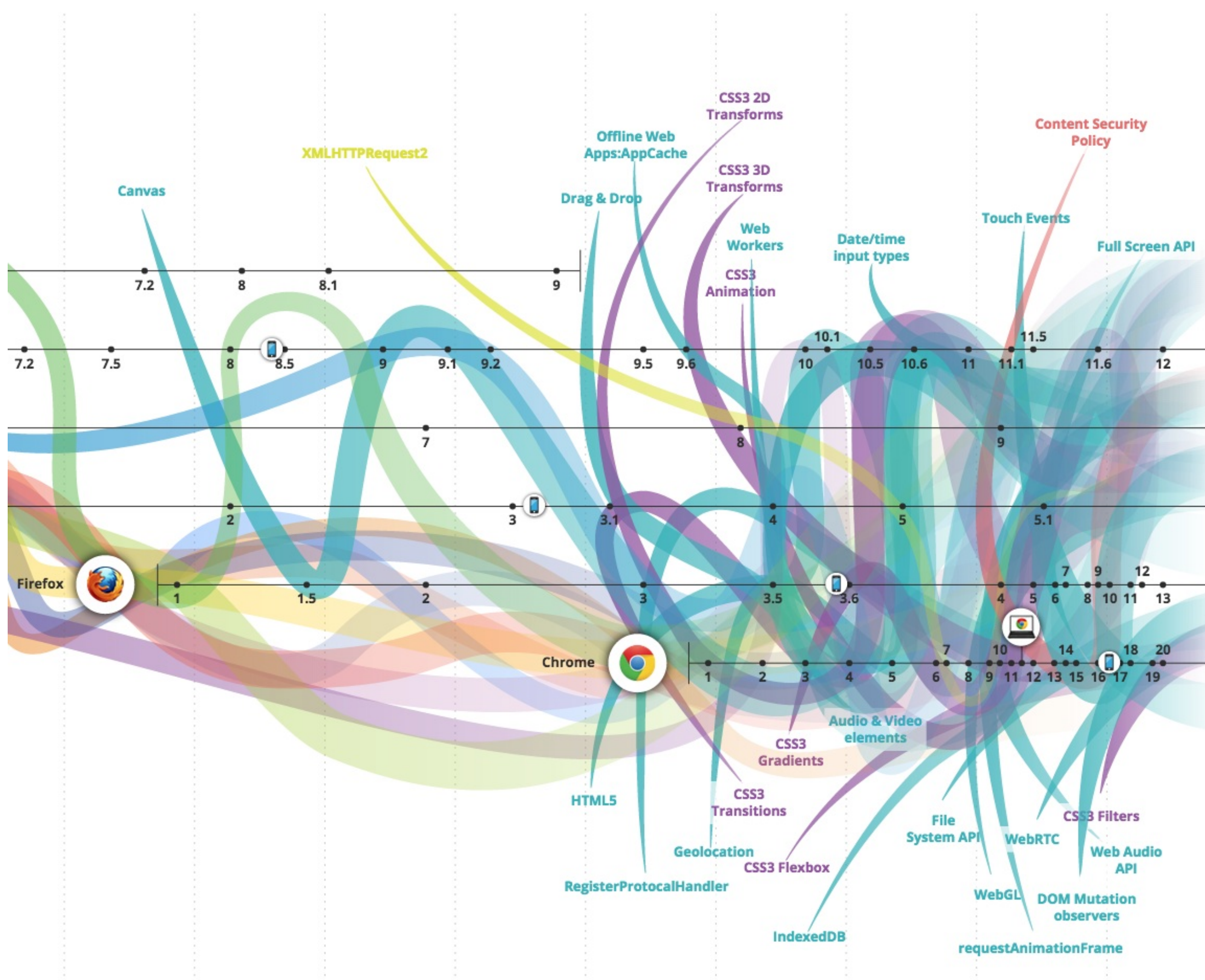
综合学习：

- UX的基础：可访问性 [观看][付费]
- 屏幕阅读器是如何支持HTML标签的 [阅读]
- Web可访问性简介 - 谷歌公开在线教育视频 [观看]
- Web可访问性简介 - WAI [阅读]
- Web应用程序的通用设计：适合所有人的Web应用程序 [阅读][付费]
- Web可访问性：入门 [观看][付费]
- 为所有人服务的网络世界 [阅读][付费]
- Web可访问性 [观看][付费]

标准/规范：

- 可访问的互联网应用程序（WAI-ARIA）现状
- 网络无障碍倡议（WAI）
- 网页内容辅助功能指南（WCAG）当前状态

学习 web／浏览器 API



图片来源*http://www.evolutionoftheweb.com/*

BOM（浏览器对象模型）和 DOM（文档对象模型）并不是唯一的通过浏览器的 web 平台可以利用的浏览器 API。不单单指 DOM 或 BOM，只要是在浏览器上编程的 API 都可以认为是一个 web 或浏览器 API (可惜的是这些 API 曾经被叫做 HTML5 API,这将它们自己的特性／标准与实际的明确 HTML5 标记语言的 HTML5 标准相混淆)。注意，web 或浏览器 API 确实包括能通过平板或手机设备上的浏览器利用的设备 API（例如，`Navigator.getBattery()`）。

你应该了解并学习那些通用的 API。让自己熟悉这些 API 的好方法之一就是去研究 HTML5test.com 上的占有率前五的浏览器 API 。

学习：

- Pro HTML5 Programming [阅读]

学习音频：

- 用 Web Audio 为你的站点添加声音 [观看]
- Web Audio 的乐趣 [观看]
- Web Audio API [阅读]

学习 **Canvas:**

- HTML5 Canvas [阅读]

注意:

MDN 上有很多关于 web/浏览器 的 API 。

- MDN Web API 参考
- MDN Web APIs API 参考-所有 API，按字母排序
- MDN WebAPI - 对应用有用的设备 API 列表

记住不是所有的 API 都在 W3C 或 WHATWG 中声明过。

除了 MDN ，下面的资料对于了解所有的 API 也是有用的：

- HTML 5 JavaScript API 索引
- HTML5 总览
- platform.html5.org

学习 JSON (JavaScript 对象表示法)

> JSON, (更加标准的说法应该是 Javascript 对象表示法)，是一个使用接近自然语言的文字来传输包含键值对的数据对象的开放标准格式。它是浏览器／服务器进行异步传输（AJAX）的时候使用的主流数据格式，在 AJAX 中基本已经替代了之前的 XML。
>
> JSON 是独立于语言的数据格式，尽管它最初起源于 Javascript 脚本语言。许多程序语言都自带解析和生成 JSON 数据的代码。
>
> Douglas Crockford 定义了最初的 JSON 格式，然而现在它的规范存在于 RFC 7159 以及 ECMA-404 这两个存在竞争关系的标准当中。其中，ECMA 标准更为轻量，它仅仅描述了允许使用的语法规则；而 RFC 的规范则在语义和安全性上进行了进一步的考虑。JSON 官方的因特网媒体类型是 application/json，文件扩展名是 .json。
>
> — 维基百科

概论:

- Javascript 对象表示法介绍: 中肯实用的 JSON 指南 [阅读][增值付费]
- json.com [阅读]
- 何为 JSON [观看]

参考文档:

- json.org [阅读]

标准／规范:

- ECMA-404 JSON 数据交换格式
- RFC 7159 Javascript 对象表示法 (JSON) 数据交换格式

架构:

- JSON API

学习 JS 模板

典型的 JavaScript 模板一般在 MV* 设计模式中，用于把 View 层（即 UI）从逻辑和数据层（比如，数据或 JSON）中分离出来。

- ES6 模板语法, Handlebars 杀手? [阅读]
- nunjucks 入门 [阅读]
- Handlebars 速成 [阅读][付费]
- Handlebars 模板 [视频][付费]
- 10 分钟掌握 Handlebars [阅读]
- Lodash 模板 [文档]

注意：

JavaScript 2015 (即 ES6) 有一个原生的模板机制叫做"模板字符串"。另外，最近跟模板有关的热词还有 JSX、"模板元素"和 HTML strings。

建议：

如果您对模板还不熟悉，首先搞定 JavaScript 的"模板字符串"。 如果项目与 React 相关，您可以使用JSX。 再往后您可以学习 nunjucks。

学习静态网页生成器

静态网站生成器通常用服务端代码来编写（例如，ruby，PHP，python，nodeJS，等），用静态文本／数据 + 模版，生成从服务器发送到客户端的静态 HTML 文件。

概论:

- 静态网站生成器 [文章]

通过JS学习计算机科学

- 六小时助你攻克四个学期的计算机课程 [视频][付费]
- Javascript 中的计算机科学 [文章]
- 使用 Javascript 编写的经典计算机科学中的范例，算法以及方法 [文章]
- Javascript 中的算法和数据结构 [视频][付费]

学习前端应用架构

综合学习：[1]

- JavaScript 应用设计 [阅读][增值付费]
- 编写 JavaScript 应用代码 [阅读]

过时的学习材料：

- 用 React 和 Ampersand 构建应用 [观看][增值付费]
- 构建现代的单页网络应用 [观看][增值付费]
- 流畅的 JavaScript: 模块化 [阅读]
- 静态应用工作指南 [阅读]
- 网络应用工作指南 [阅读]
- 前端参考问卷 [阅读]
- 对人类友好的 JavaScript [阅读]
- Nicholas Zakas: 可扩展的 JavaScript 应用的构建方法 [观看]
- JavaScript 功能组织 [观看][增值付费]
- 构建大型 Javascript 应用 [阅读]
- 绝无仅有地出色 [阅读]
- UI 架构 [观看][增值付费]
- 网络 UI 架构 [观看][增值付费]

注释：

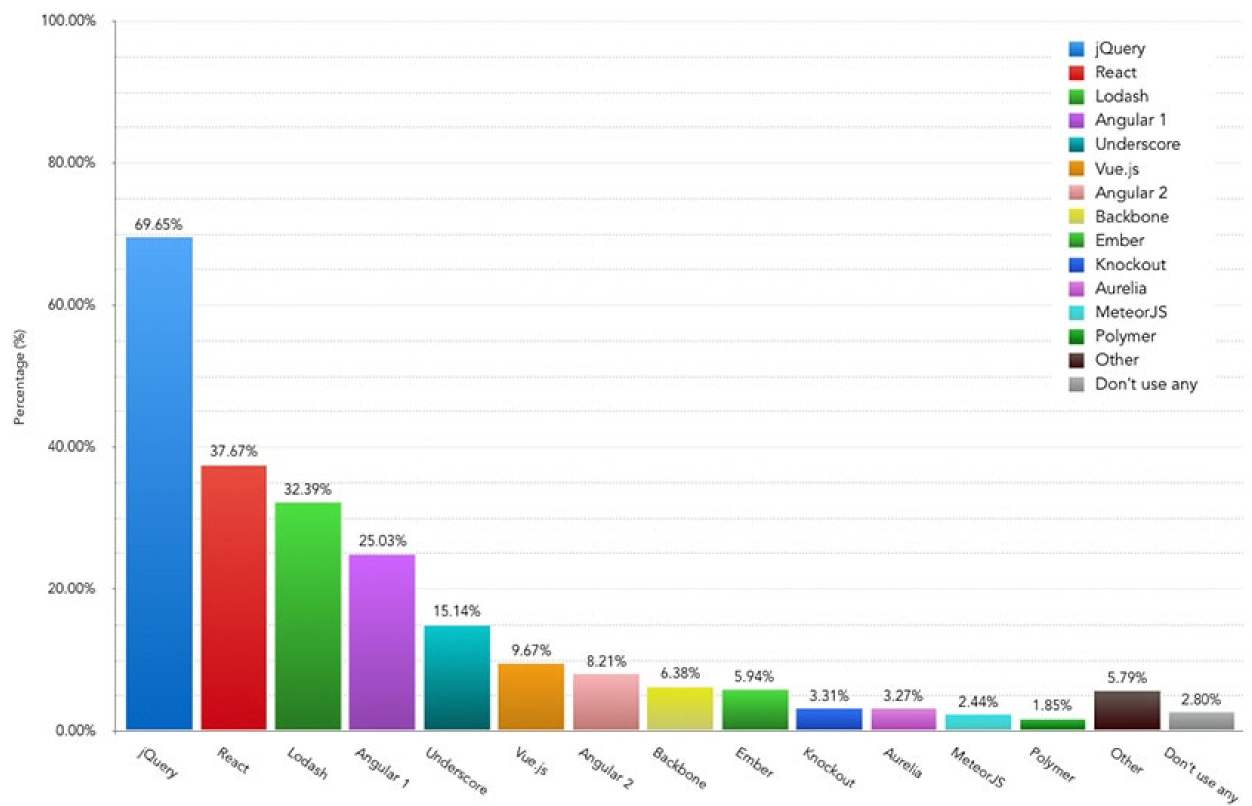
这个主题下面并没有太多新近产出的内容。大部分可供学习如何构建前端／单页／Javascript 应用的内容都建立在某个具体工具之上，例如 Angular，Ember，React 或者 Aurelia。

建议：

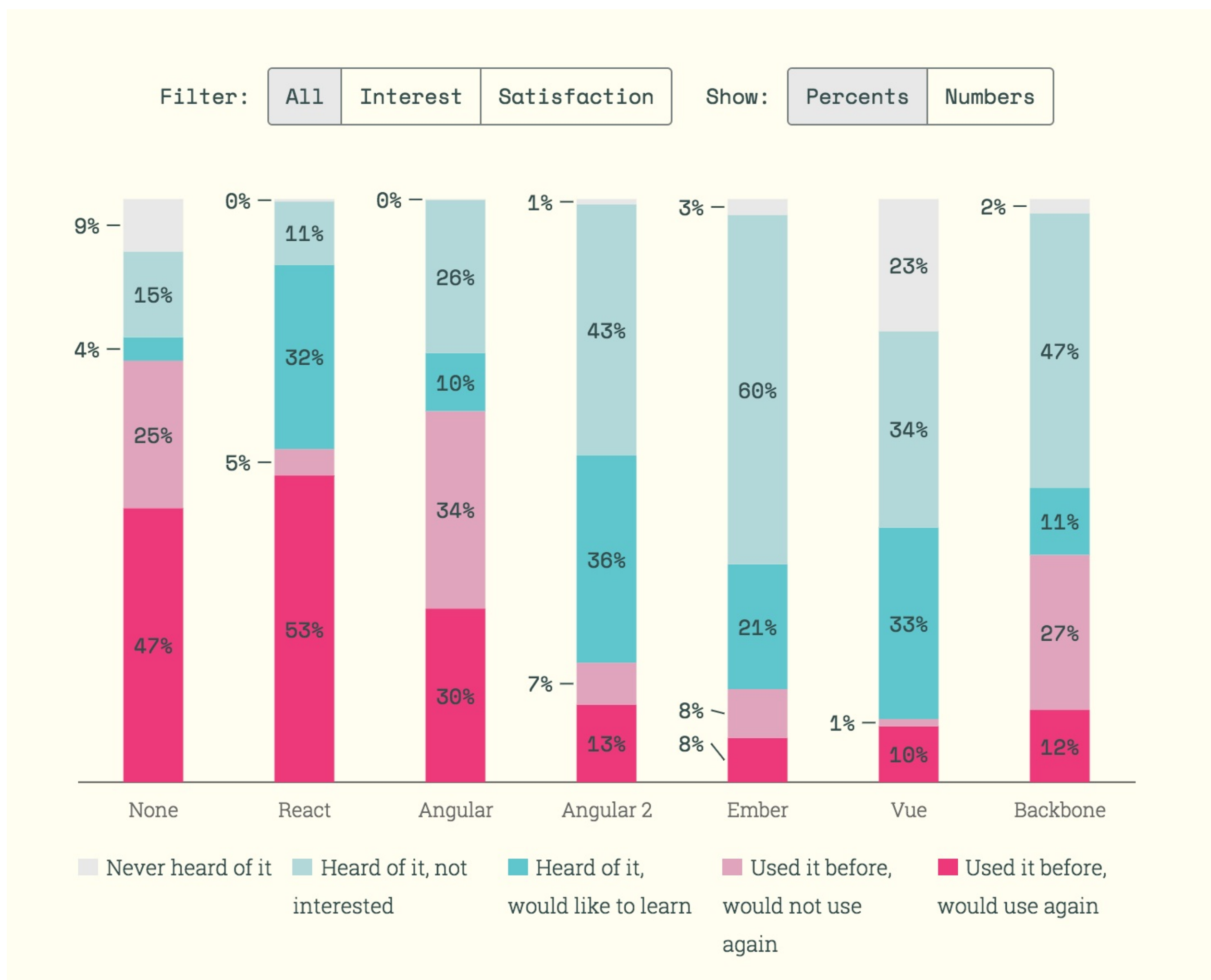
[1] 2017 年，请学习 Webpack, React, 以及 Redux，并且请从 "A Complete Intro to React" 和 "Building Applications with React and Redux in ES6"开始。

问卷调查结果：

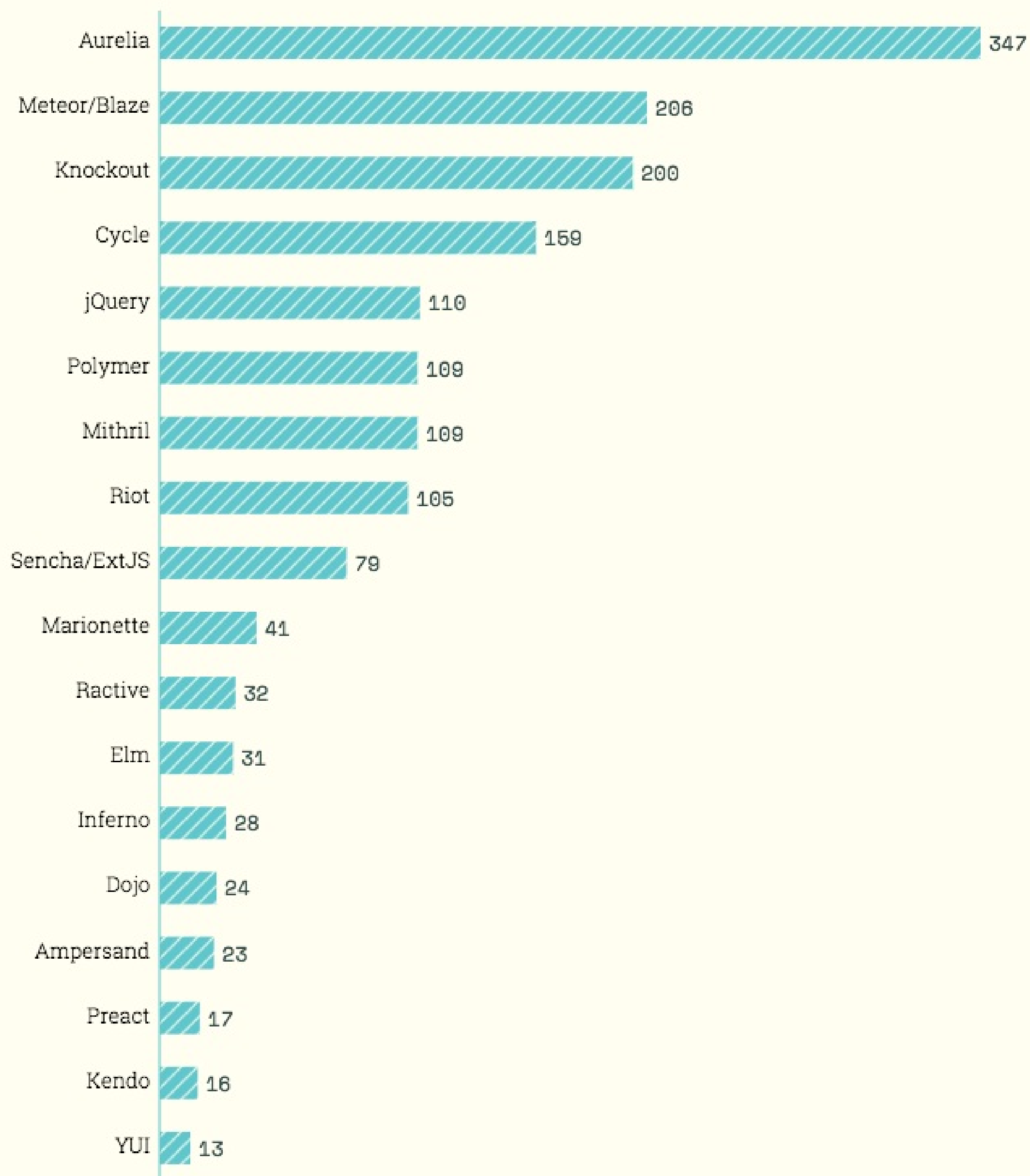
下图来自 2016 前端工具调查 (4715 名开发者参与) 以及 2016 JS 现状调查 (9307 名开发者参与)



图片来源: *https://ashleynolan.co.uk/blog/frontend-tooling-survey-2016-results*



图片来源: *http://stateofjs.com/*



图片来源: *http://stateofjs.com/*

学习数据（例如，JSON）API 的设计

- Node.js 中的 API 设计 (使用 Express & Mongo) [视频][付费]
- 构建不恶心人的 APIs [文章][付费]
- JSON API [文章]
- Node.js 中的 RESTful Web API 设计 - 第二版 [文章][付费]

学习 React & Redux

React:

- React.js Introduction For People Who Know Just Enough jQuery To Get By [阅读]
- React.js Fundamentals [视频]
- 13 things you need to know about React [阅读]
- Tutorial: Intro To React [阅读]
- React Enlightenment [阅读]
- ReactJS For Stupid People [阅读]
- REACT FOR BEGINNERS [视频]
- Complete Introduction to React (feat. Redux and React Router) [视频]
- React In-depth: An exploration of UI development [阅读]
- Complete Intro to React v2 (feat. Router v4 and Redux) [视频][$]
    - Welcome to A Complete Intro to React [阅读]
- Build Your First Production Quality React App [视频][付费]

Redux:

- You Might Not Need Redux
- A Dummy's Guide to Redux and Thunk in React [阅读]
- Redux Tutorials [视频]
- Getting Started with Redux [视频]
    - https://github.com/dwyl/learn-redux/blob/master/egghead.io_video_tutorial_notes.md
- Learn Redux [视频]
- 10 Tips for Better Redux Architecture [视频]
- Building React Applications with Idiomatic Redux [视频][付费]

注意:

一旦学会了 React 你可能会考虑了解一下 Preact 和 Inferno。

当你掌握了 Redux 看一看 MobX 或者从零开始自己实现一套类 Redux 的方法。

学习渐进式 Web 应用

> 不同于传统应用，渐进式 web 应用是普通网页（或网站）与移动应用的混合体。这种新的应用模型尝试将移动端的体验优势与现代浏览器提供的大部分特性相结合。
>
> 在2015年，设计师 Frances Berriman 和 Google Chrome 工程师 Alex Russell 创造了“渐进式 Web 应用”这个术语，这个词用来形容那些充分利用现代浏览器新特性的应用，这些特性包括 Service Workers 和 Web App Manifests，它们使得 web 应用一跃成为用户原生系统中应用的头等公民。
>
> 按照 Google Developers 网站的说法，（渐进式 Web 应用）包含以下特征：
>
> - 渐进式 - （应用）对每个用户都可用，（开发者）不必关心用户选择何种浏览器。因为应用是以渐进增强作为核心原则构建的。
> - 响应式 - 适应于任何形式的设备：桌面、移动、平板或者还未出现的设备形式。
> - 独立的连接 - Service workers 允许应用离线工作，或者在低质量的网络环境下工作。
> - 类应用 - 用户感觉像是原生应用，因为它们拥有着原生风格的交互和导航。
> - 保持最新 - 得益于 service worker 的更新进程，应用总是能够及时更新。
> - 安全 - 服务构建于 HTTPS 上，从而能够防止被嗅探，并确保内容不被篡改。
> - 可被发现的 - 能够被识别为“应用”且允许被（本地应用）搜索引擎找到，这要多亏了 W3C 的 manifests[6] 和 service worker 的注册作用域。
> - 可再次唤起 - 经由一些特性（比如推送通知），让再次唤起变得更简单。
> - 可安装 - 用户可以无需借助 app store，而直接将他们觉得最有用的应用“保留”在屏幕首页上。
> - 可链接 - 可以通过 URL 进行分享，无需复杂的安装过程。
>
> — 维基百科

- 渐进式 Web 应用 [阅读]
- 新手的渐进式 Web 应用指南 [阅读]
- 渐进式 Web 应用 [阅读]
- 从渐进式 Web 应用起步 [观看][$]
- 构建一个渐进式 Web 应用 [观看][$]
- Google 渐进式 Web 应用介绍 [观看]
- 原生应用要完蛋了 [观看]
- 为什么原生应用真的要完了：原生应用要完 第2部分 [阅读]

学习设计 JS API

- 设计更好的 Javascript APIs [文章]
- 编写 Javascript APIs [文章]

学习web开发工具

> Web开发工具可以让web开发者们对自己的代码进行测试和调试。这些工具并不支持直观的创建web页面，也使得它们与站点集成工具和IDEs工具不一样, 更确切的说，它们是用来调试开发者所面对的web网站或web应用接口的工具。
>
> Web开发工具一般是浏览器的附加软件或内置在web浏览器中。时下最受欢迎的浏览器如：谷歌浏览器, 火狐浏览器, 欧朋浏览器, IE浏览器, 以及苹果浏览器，它们都为web开发者内置了开发工具，在各自的插件下载中心还有很多其他的插件可以找到。
>
> Web开发工具使得web开发者可以在工作中应用多样化的技术，包括html、css、Dom、JavaScript以及web浏览器提供的其它组件。由于web浏览器越来越流行，web浏览器已经为开发人员包含了更多的功能。
>
> — *Wikipedia*

虽然大多数浏览器已经装备了web开发工具，但谷歌开发工具是目前最受关注和广泛使用的工具。

建议您学习和使用谷歌网页开发工具，只因为学习网页开发人员工具的最佳资源皆围绕着谷歌开发工具。

学习谷歌**web**开发工具:

- 谷歌开发者工具 [阅读][付费]
- 探索学习谷歌开发工具
- 掌握谷歌开发工具 [阅读][付费]
- 使用Chrome开发者工具 [阅读][付费]

谷歌**web**开发工具相应文档:

- 命令行API参考
- 键盘和UI快捷键参考
- 各开发工具文档
- 自定义开发工具

新闻/时讯/播客/小贴士:

- 开发小贴士

学习命令行的使用

> 一个命令行界面或者命令语言解释程序（CLI），也被称为命令行用户界面，控制台用户界面，以及字符用户界面（CUI）。用户通过在命令行界面中连续地键入文字（命令行）向计算机传达命令，从而与计算机程序进行互动。
>
> — 维基百科

基础:

- Bash指南 [文章]
- Codecademy: 学习使用命令行 [视频]
- 命令行高级用户 [视频]
- 成为危险的命令行使用者 [文章] [免费增值]
- 遇见命令行 [文章][付费]

进阶:

- 高级命令行技术 [视频][付费]

学习 Node.js

> Node.js 是一个可用于开发服务器端 Web 应用的开源跨平台运行时环境。Node.js 应用是由 JavaScript 语言实现的 ，可以运行在 OS X、Windows、Linux、FreeBSD、NonStop、IBM AIX、IBM z 和 IBM i 系统的 Node.js 环境中。 Node.js 由 Linux 基金会的合作伙伴 Node.js 基金会所有并提供支持。
>
> Node.js 采用基于事件驱动的架构和非阻塞的 I/O API 设计，以优化即时 Web 应用的吞吐量和可扩展性。Node.js 使用 Google 的 V8 Javascript 引擎执行代码，并且绝大部分的基础模块是用 Javascript 写的。 Node.js 包含了一个内建的库， 即使没有传统的服务器软件如：Apache , Nginx 或者 IIS 您也可以很方便地使用 Node.js 创建一个 Web 服务器。
>
> — 维基百科

综合学习:

- Node 编程艺术 [阅读]
- Node.js 入门 [视频][付费]
- Evented Mind 带你 Node.js 入门 [视频][付费]
- io.js 和 Node.js 入门 [视频][付费]
- 学习 Node 服务器端编程 [阅读][付费]
- 带你学习 Node.js [自行研讨]
- Node.js 基础 [视频][付费]
- Node.js 实战 [阅读][付费]
- 使用 Node.js 开发实时 Web 应用 [视频]
- 使用 Node.js 开发亚马逊 Web 服务从零到上线 [视频][付费]

关于模块的学习

概论：

- jsmodules.io
- 深入理解 ES6 模块 [read]
- 探索 JS —— 模块 [read]

参考文献：

- MDN - export
- MDN - import

注释：

关于浏览器加载模块的部分，我们仍在寻求支持，在那之前请参考 "ES Module Loader Polyfill" 以及 "JavaScript Loader Standard"。

关于模块加载和打包工具

Webpack:

- Webpack
- 深入 Webpack [文章]
- Webpack 基础知识 [视频][付费]
- Survivejs.com Webpack Book [文章]

Rollup:

- Rollup

SystemJS:

- 用 SystemJS 和 jspm 编写现代的，模块化的 JavaScript 代码 [视频][付费]

注释:

开发者使用类似于 Gulp 之类的 JS 模块化打包工具来帮助开发并不鲜见。然而，许多的 Gulp 插件本质上仅仅是利用了 Webpack, Rollup, 或者 SystemJS 而已。

关于包管理工具的学习

> 包管理工具或者包管理系统，是一些以一致的风格对计算机操作系统中的软件包进行自动化安装、升级、配置以及删除的软件工具。它通常维护着一个关于软件依赖性和版本信息的数据库，用以解决软件之间的配对和依赖问题。
>
> — 维基百科

综合学习：

- Javascript 包管理工具的工作原理入门
- npm 和 Bower 中神奇的 SemVer Ranges [阅读]
- 包管理工具：给前端开发新手的介绍性指南 [阅读]
- npm 文档
- yarn 文档

学习版本管理

> 版本管理做为软件配置管理的要素，也被称为修定管理或者源代码管理。它用于管理文档，程序，大型网站，以及其它信息集合的变更。变更使用一串数字或者字母来加以标识，这个标识被称做『修订号』『修订标识』或简称『修订』。比如说，一些文件的初始状态是『修订1』当第一个改变发生后，这些文件变成了『修订2』如此继续。每一次修订都和一个时间邮戳及修订的人有关。所有的『修订』都可以被比较，复原，以及针对特定类型的文件（译者注：此处指文本文件）进行合并。
>
> — 维基百科

当下最流行的版本管理方案是 Git。 学习它！

综合学习：

- codeschool.com [交流]
- 正确使用 Git [阅读]
- Git 基础 [观看][收费]
- 全面学习 Git [阅读]
- Ry的 Git 入门 [阅读]

高级内容：

- 高级 Git 教程 [阅读]
- 专业 Git [阅读]
- 学习 Git 分支 [交流]

参考文档：

- https://git-scm.com/doc

学习构建及任务自动化技术

> 构建自动化是对软件构建及相关流程自动化的过程。这些相关流程包括：将计算机源码编译成二进制代码，打包二进制代码以及运行自动化测试。
>
> — 维基百科

常规学习:

- Gulp 入门 [阅读][收费]
- Gulp 基础 [观看][收费]
- 使用 Gulp.js 自动化构建 JavaScript [观看][收费]

参考和文档:

- Gulp

建议:

虽然 Gulp 很棒，但你可能只需要 `npm run` 命令。在把额外复杂度添加到你的应用程序技术栈之前，先问问自己：仅靠 `npm run` 能否完成任务？如果答案为否，那就使用 Gulp。

阅读:

- Grunt 滚蛋! 使用 npm 来当构建工具的指南
- 如何使用 npm 来当构建工具
- 使用 npm run 自动化任务
- 在你的下一个项目里用 npm 当构建系统
- 用 npm 作为任务运行启动器 [观看][收费]
- 为什么我舍弃 Gulp 和 Grunt 转而使用 npm 脚本？
- 为什么使用 npm 脚本？

学习网站性能优化

> 页面性能优化，又称 WPO，亦或是站点优化，它是一门提升页面在用户 Web 浏览器中下载及展现速度的学问。随着全球平均网速的不断提升，对于网站管理员及站长们来说，关注用户所访问网站的渲染时间是很有必要的。
>
> — 维基百科

通用知识学习：

- 浏览器渲染优化 [观看]
- 更快的网站：针对 Web 开发者的最佳性能实践 [阅读][$]
- 高性能 Web 站点：前端工程师必知必会 [阅读][$]
- JavaScript 的性能基石 [阅读]
- PageSpeed Insights 规则 [阅读]
- perf-tooling.today [阅读]
- 性能日历 [阅读]
- perf.rocks [阅读]
- 使用 WebPageTest [阅读][$]
- Web 性能日记 卷 2 [阅读][$]
- Web 性能: 权威指南 [阅读]
- 网站性能 [观看][$]
- 网站性能优化 [观看]

学习测试

> 单元测试 - 编写程序时，单元测试是一种测试方法。它使用特定的数据，特定的手段，特定的操作，以检验独立的一段代码，一个或多个模块是否能正常工作。你可以将『单元』看做是应用程序最小可测试的单位。
>
> — 维基百科
>
> 功能测试 - 功能测试是一种质量保证（QA）的过程。同时它也是一种黑盒测试，它的测试用例基于组件的设计。功能测试即给定输入检查输出，它很少考虑程序的内部结构（不像白盒测试）。功能测试通常描述的是系统做什么。
>
> — 维基百科
>
> 集成测试 - 集成测试（有时被称为集合和测试，简称 I&T ）是软件测试的一种过程。在这个过程中软件的独立模块被整合后做为一个组一起测试。它发生在单元测试后，确认测试前。集成测试把经过单元测试的模块聚合成一个整体进行测试，并做为系统测试中系统集成是否成功的结果。
>
> — 维基百科

综合学习：

- 前端优先: 测试并原型化 JavaScript 应用 [视频][付费]
- 编写测试驱动的 JavaScript [视频][付费]
- JavaScript 测试 [视频]
- JavaScript 测试妙方 [阅读][付费]
- 可测试的 JavaScript [阅读][付费]
- 测试驱动的 JavaScript 应用: 开发快速, 自信, 可维护的代码[阅读][付费]
- 测试驱动的 JavaScript 开发 [阅读][付费]
- Web 测试之路: 新手自动测试指南 [阅读][付费]

学习无头浏览器

> 无头浏览器是没有用户图形界面的浏览器。
>
> 无头浏览器提供了对于网页的自动控制，这种控制所依赖的环境和现代 web 浏览器类似，然而却是通过命令行界面或者网络通讯实施。它们是良好的网页测试工具，提供了和普通 web 浏览器同样的功能，包括渲染 HTML、页面布局、颜色以及字体，同时也提供了 JavaScript 和 AJAX 的加载和执行，这是其它测试工具通常做不到的。Google 在 2009 年的时候声称，使用无头浏览器可以帮助他们的搜索引擎在网站中定位那些使用 AJAX 技术加载的内容。
>
> — 维基百科

- 使用 PhantomJS 和 CasperJS 使网络自动化 [视频][增值付费]
- PhantomJS 入门 [文章][增值付费]
- PhantomJS 教材 [文章][增值付费]
- PhantomJS 网络自动化 [视频]
- 快捷的 PhantomJS [视频][增值付费]

学习离线开发

所谓的离线开发（又称离线优先）是围绕某些设备在开发实践所中涉及的知识和讨论，这些设备并非总是处于连接到互联网或者电源。

概论:

- 创造 HTML5 离线网络应用 [文章]
- 关于创造离线网络应用，你应该知道的事情 [文章]
- 离线优先 [文章]
- offlinefirst.org [文章]
- 你的首个离线 Web 应用 [文章]

关于网络／浏览器／应用的安全性的学习

- 浏览器安全性手册 [文章]
- 前端安全性能 [视频]
- Hacksplaining [文章]
- HTML5 安全性参考手册 [文章]
- HTTP 安全性最佳实践 [文章]
- 网络开发中的身份和数据安全性: 最佳实践 文章
- 给网络开发者的安全性建议: 采用 JavaScript, HTML, 和 CSS [文章][付费]
- 网络应用安全性基础 [文章]
- 因特网：加密和公钥 [视频]
- 因特网：解密和犯罪 [视频]
- 错综复杂的网络世界：现代网络应用安全性指导手册 [文章][付费]
- 网络安全性基础 [文章]
- 网络安全性 [文章]

多平台开发学习



图片来源: http://bradfrost.com/blog/post/this-is-the-web/

网站与网络应用总是能够在种类繁多的个人电脑，手提电脑，平板电脑和手机中运行。除此以外，许多的新型设备如手表，恒温器，冰箱等也开始支持同样的功能。决定支持怎样的设备，以及怎样进行开发来支持这些设备的过程，我们称之为“多设备开发策略”。下面列出的是最常见的多设备开发策略。

- 为所有的设备搭建一个响应式（RWD）网站／应用。
- 为所有的设备搭建一个自适应渐进增强 的网站／应用。
- 为每个不同设备或设备群体搭建网站，网络应用，或者混合模式的移动应用。
- 采用以上 1，2，3 点中的某些方式，重构已存在的项目。

概述:

- 一分为二（上）- 响应式网站设计 [文章][$]
- 一分为二（下）- 适用于所有设备的设计方案 [文章][$]
- 自适应网站设计 [文章][$]
- 符合渐进增强原则的设计 [文章][$]
- 移动端网站开发 [视频]
- 响应式 HTML 邮件设计 [视频][$]
- 响应式图片 [视频]
- 响应式字体 [视频][$]
- 响应式网站设计 [视频][$]
- 响应式网站设计基础 [视频]

响应式内部通讯，新闻或者播客网站:

- 响应式播客网站设计
- 响应式内部通讯网站设计

导向学习

这个部分是关于通过学校，课程，项目以及训练营来进行导向学习。

培训学习

下面的表格列举了一些通过付费来指导你学习的前端课程，项目、学校和训练营。

如果觉得付费指导学习比较昂贵，你也可以使用屏幕录像，书籍，文章等途径来进行自我驱动型的前端开发学习。

机构	课程	价格	现场教学	是否支持远程
Betamore	Web 前端开发	8,500	巴尔的摩（译者注：美国马里兰州最大的城市）	
BLOC	成为一名前端开发者	4,999		是
DecodeMTL	学习 Web 前端开发	2,500	蒙特利尔（译者注：加拿大东南部港市）	
The Flatiron School	Web 前端开发入门	3,500	纽约（美国）	
General Assembly	Web 前端开发	3,500	多个地点	
HackerYou	沉浸式 Web 前端开发	7,000 - 7,910	多伦多（加拿大）	
Iron Yard	Front End Engineering	12,000	多个地点	
The New York Code + Design Academy	前端 101	2,000	纽约（美国）	
Thinkful	Web 前端开发	300 每月		是
Turing School of Software & Design	前端工程化	20,000		是
Udacity	Front-End Web Developer Nanodegree	200 每月	多个地点	是

前端开发学习的起点

那种需要跟随单一个体去学习前端技术的想法逐渐变得毫无意义。前端技术先进实践者们正在使一种新的学习方式成为可能：他们在前端社区里创造出了足够多的内容，使您能通过关注前端“新闻”媒体(通过 新闻通讯, 新闻, & 播客)轻松追随社区／领导者，从而进行学习。

前端资讯、新闻站和播客

通用前端资讯、新闻和播客：

- 超级 Web 秀
- 开发小帖士
- 前端开发周报
- 前端欢乐时光
- 5 分钟前端新闻
- 前端前沿
- 前端关注
- 前端资讯
- 前端博客
- 移动 Web 周报
- 不换行空格秀
- 开源 Web 平台每日文摘
- 行话秀
- UX 设计资讯
- 变形秀 - SitePoint
- Web 前瞻
- Web 开发阅读清单
- Web 平台播客
- Web 工具周报

HTML/CSS 资讯：

- 可访问性周报
- CSS 周报
- 响应式设计周报

JavaScript 资讯、新闻和播客：

- JS 回声
- ECMAScript 日报
- ES.next 新闻
- FiveJS
- JavaScript 播送
- 闲谈 JavaScript
- JavaScript Kicks
- JavaScript 在线

- JavaScript 周报
- JavaScript.com
- React 动态

图形和动画资讯、新闻和播客：

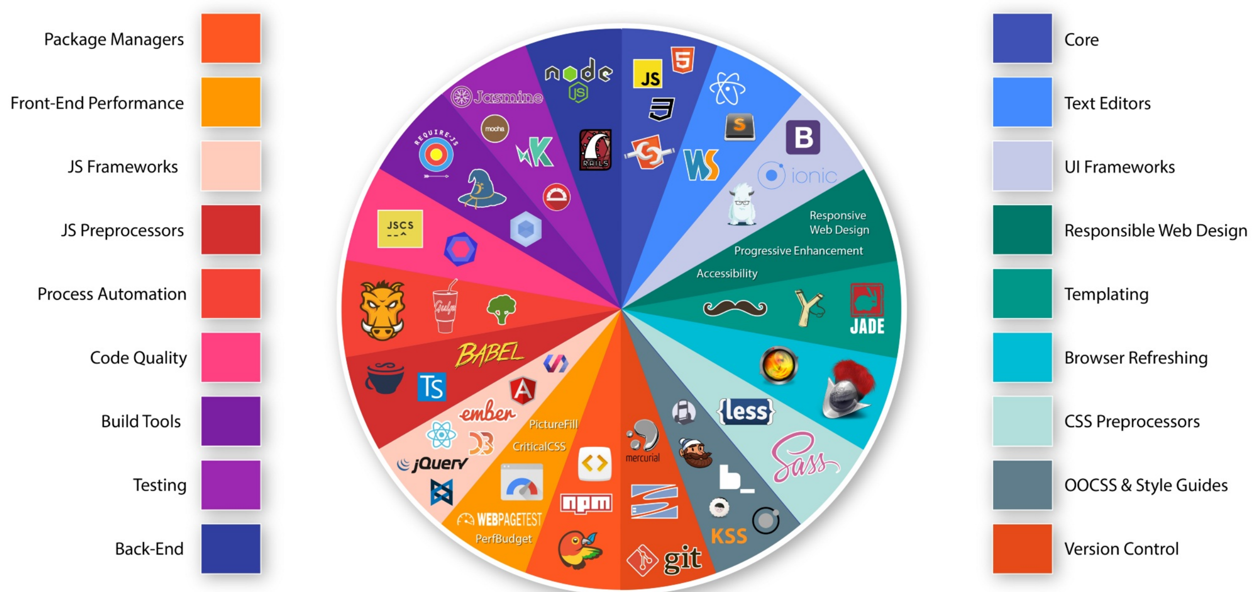
- 运动及意义
- 响应式图片社区小组资讯
- SVG 沉浸式播客
- Web 动效周报

第三部分：前端开发工具

第三部分简要地介绍和罗列了一些前端圈内的工具。

在学习这些工具之前，请确保你理解了它们的分类，分类内包含了某一系列的工具。

需要注意的是，某个工具出现在列表中或者某类工具出现在文档中，并不代表我个人主张一名前端开发人员应该学习和使用它。请选择你自己的工具箱。我只是提供了常见的选项。



图片来源: *https://medium.com/@withinsight1/the-front-end-spectrum-c0f30998c9f0*

Doc/API 浏览工具

浏览常用开发者文档和 API 参考资料的工具。

- Dash [OS X, iOS][付费]
- DevDocs
- Velocity [Windows][付费]
- Zeal [Windows, Linux]

SEO 工具

- 关键词工具
- Google 站长工具
- Varvy SEO 工具

寻找 **SEO** 工具的工具:

- SEO 工具 —— 完整列表

原型设计和线框图工具

设计工具:

- Axure [付费]
- Balsamiq Mockups [付费]
- Justinmind [付费]
- UXPin [免费增值]

协作 / 展示工具:

- InVision [免费增值]
- Conceptboard [免费增值]
- myBalsamiq [付费]

制图工具

- draw.io [免费增值]
- Cacoo [免费增值]
- gliffy [免费增值]

HTTP / 网络工具

- Charles [付费]
- Chrome 开发者工具网络面板
- Insomnia [免费增值]
- Paw [付费]
- Postman [免费增值]

代码编辑工具

> 源代码编辑器是为编辑计算机程序源代码而设计的文字编辑程序。它可能是一个单独的应用程序，也可能嵌入在一个集成的开发环境（IDE）或者浏览器中。因为编程人员最基本的工作就是编写源码，因此源代码编辑器是最基本的编程工具。
>
> — 维基百科

使用诸如 Notepad 和 TextEdit 之类简单的文字编辑器即可满足前端代码编写的最低要求。然而，大多数的前端开发者倾向于使用为编辑程序语言而专门设计的代码编辑器。

代码编辑器有各种各样的类型和大小，因此完全可以根据个人偏好来进行选择。做出选择以后，全面地学习使用方法，然后投入到 HTML，CSS，DOM 和 Javascript 的学习中去吧。

尽管如此，我还是强烈认为一款合格的代码编辑器至少应该达到以下标准（在默认设置或者使用插件的情况下）：

1. 能够找到关于如何使用该编辑器的条理清晰的文档；
2. 能够对 HTML，CSS 以及 Javascript 代码的质量进行汇报（提示／诊断／报错）；
3. 提供 HTML，CSS 以及 Javascript 代码高亮；
4. 提供 HTML，CSS 以及 Javascript 代码补全；
5. 可以利用插件进行自定义构建；
6. 有许多可以使用的第三方／社区插件，可以用来把编辑器自定义为自己喜欢的样子；
7. 小而轻量，同代码不相耦合。

代码编辑器：[1]

- Atom
- Brackets
- Sublime Text [付费]
- WebStorm [付费]
- Visual Studio Code

在线代码编辑器：

- Cloud9 [增值付费]
- Codeanywhere [增值付费]

可运行，可分享的代码编辑器：

可以用来分享一定数量，且即刻可以运行在 web 浏览器中的代码。这类工具并非真正的代码编辑器。

- CodePen [增值付费]
- jsbin.com [增值付费]
- jsfiddle.net
- liveweave.com
- Plunker

建议：

1 强烈建议使用 Visual Studio Code ，这款编辑器非常优质并且处于持续的更新中，另，作为微软出品的软件，其质量和更新速度都能够到有效的保证。

浏览器上的神兵利器

浏览器环境下 **js** 编码的实用工具:

- History.js
- html2canvas
- Platform.js
- URI.js

判断某浏览器是否支持某功能的常见参考工具:

- 查看浏览器对 image 标签各属性的支持情况
- Browserscope
- caniuse.com
- Firefox 平台状态 - Web平台功能的实施和标准化路线图
- HTML5 Please
- HTML5 测试
- iwanttouse.com
- jscc.info
- 平台状态
- whatwebcando.today

浏览器开发/调试工具:

- 谷歌开发工具(别名 DevTools)
    - 各开发工具文档
    - 命令行API参考
    - 键盘和UI快捷键参考
    - 设置
- 火狐开发工具
- IE 开发工具 (别名 F12 工具)
- Safariweb 检查器
- Vorlon.js

判断某浏览器是否支持某功能的浏览器编码工具:

- Feature.js
- Modernizr

多样的浏览器**Polyfills/Shims:**

- console-polyfill
- HTML5 Cross Browser Polyfills
- fetch
- socket.io
- SockJS
- webcomponents.js
- webshim

浏览器托管测试、托管自动化:

- Browserling [免费 收费]
- BrowserStack [收费]
- CrossBrowserTesting.com [收费]
- Nightcloud.io
- Sauce Labs [收费]

无界面浏览器:

- PhantomJS
    - PhantomCSS
- slimerjs
- TrifleJS
- Zombie.js

浏览器自动化:

应用于功能测试和 monkey 测试（搞怪测试）.

- CasperJS
- Nightmare
- TestCafe

浏览器上的奇淫巧技:

- browserhacks.com

HTML 工具

HTML 模板/样板/入门套件：

- dCodes
- 电子邮件样板
- HTML5 样板
- HTML5 骨架
- 移动端样板
- Web 入门套件样板及多设备开发工具

HTML 兼容库：

- html5shiv

转译语言（**Transpiling**）：

- HAML
- Pug
- Markdown

参考：

- 元素属性
- 元素
- HTML Arrows
- HTML 实体字符速查
- HTML 编程接口浏览器支持速查
- htmlreference.io

检查/提示：

- HTMLHint
- html-inspector

优化：

- HTML Minifier

在线创作/生成/试验工具：

- tablesgenerator.com

代码书写公约：

- HTML 代码指南
- 一致且符合规范的 HTML 书写原则

工作流：

- Emmet

HTML 大纲：

- HTML 5 Outliner

当月 **GitHub** 热门 **HTML** 仓库：

https://github.com/trending?l=html&since=monthly

CSS 工具集

桌面和移动端 **CSS** 框架：

- Base
- Basscss
- Bulma
- Bootstrap 3 或者 Bootstrap 4
- Concise
- Foundation
- Material Design Lite (MDL)
- Metro UI
- Picnic
- Pure.css
- Semantic UI
- Skeleton
- Spectre.css
- tachyons

移动专属 **CSS** 框架：

- Ratchet

CSS 重置**:**

> css 重置文件（或者是 "Reset CSS"）是简短的，经常压缩的（最小化的）一组 css 规则集合。这些规则将所有 HTML 元素的样式重置为统一的基准。
>
> — *cssreset.com*

- Eric Meyer 的 "Reset CSS" 2.0
- Normalize

转译工具：

- pleeease.io
- PostCSS 和 cssnext
- rework 和 myth
- Sass/SCSS
- Stylus

参考文献**:**

- CSS 光标
- css3test.com
- css3clickchart.com
- cssreference.io
- CSS 索引 - 由 CSS 标准定义的一个术语列表
- css4-selectors.com
- css4 Rocks
- 触发 CSS... 布局游戏，绘画和合成
- CSS 技巧年鉴
- cssvalues.com
- MDN CSS 参考

检查/提示:

- CSS Lint
- stylelint

代码格式器/美化器:

- CSScomb
- CSSfmt

优化工具:

- clear-css
- cssnano
- CSSO

在线创建/生成/实验工具集:

- 制作你的 CSS 箭头
- CSS Matic
- 享受 CSS
- 玩转 Flexbox
- flexplorer
- patternify.com
- patternizer.com
- 终极 CSS 渐变生成器

CSS 架构设计:

- Atomic Design [阅读]
- BEM
- ITCSS

- OOCSS [阅读]
- SMACSS [阅读][$]
- 适用于 CSS 的可扩展的模块化结构(SMACSS) [视频][收费]
- SUIT CSS
- rscss

编程/架构设计规范:

- CSS 编码指南 [阅读]
- css-architecture [阅读]
- cssguidelin.es [阅读]
- 常用的 CSS [阅读]
- 可维护的 CSS [阅读]
- 灵活开发的标准，可持续维护的 HTML and CSS [阅读]
- Airbnb CSS / Sass 格式指南 [阅读]

本月 **GitHub** 上的热门项目：

https://github.com/trending?l=css&since=monthly

DOM 工具

DOM 类库/框架:

- Bliss
- jQuery
  - You Don't Need jQuery
- Zepto
- cash
- Umbrella JS

DOM 工具库:

- Keypress
- Tether
- clipboard.js

DOM 事件工具:

- Keyboard Event Viewer

DOM 性能工具:

- Chrome DevTools Timeline
- DOM Monster

参考文献:

- 事件
- DOM 浏览器支持
- DOM 事件浏览器支持
- HTML 用户界面浏览器支持
- MDN 文档对象模型 (DOM)
- MDN 文档对象模型
- MDN 文档对象模型
- MDN 事件参考资料
- MSDN 文档对象模型 (DOM)

DOM Polyfills / Shims:

- dom-shims
- 鼠标事件 Polyfill: Web 平台统一事件系统

虚拟 DOM:

- jsdom
- virtual-dom

JavaScript 工具

JS 实用工具：

- accounting.js
- async
- axios
- chance
- date-fns
- format.js
- immutable
- is.js
- lodash
    - 舍弃 Lodash、Underscore
- Math.js
- Moment.js
- Numeral.js
- string.js
- underscore.js
    - 舍弃 Lodash、Underscore
- voca
- wait
- xregexp.com

转译 / 类型检测 (ES *to ES*)：

- Babel
- TypeScript
- Flow

代码解析引擎：

- Tern

JavaScript 兼容性检查器：

- jscc.info/

Linting/Hinting & Style Linter：

- eslint

单元测试：

- AVA
- Jasmine
- Mocha
- Tape

单元测试之断言：

- Chai
- expect.js
- should.js

单元测试之测试间谍（**Test Spies**）、存根（**Stubs**）、模拟（**Mocks**）：

- sinon.js
- Kakapo.js

代码格式化／美化工具：

- esformatter
- js-beautify
- jsfmt
- prettier

性能测试：

- benchmark.js
- jsperf.co

可视化、静态分析、复杂度、覆盖率工具：

- Coveralls [$]
- Esprima
- istanbul

优化工具:

- UglifyJS 2
- optimize-js

代码混淆:

- Javascript Obfuscator [free to $]
- JScrambler [$]

可共享／运行的代码编辑器：

- es6fiddle.net
- jsbin.com [free to $]
- jsfiddle.net

在线正则表达式编辑器／可视工具：

- debuggex
- regex101
- regexper
- RegExr

著作约定工具:

- Airbnb's ESLint config, following our styleguide
- Standard - ESLint Shareable Config

本月 **GitHub** 上的热门 **JS** 仓库：

https://github.com/trending?l=javascript&since=monthly

依赖度最高的 **NPM** 包：

https://www.npmjs.com/browse/depended

静态网页构建工具

网页构建工具列表: [1]

- staticgen.com
- staticsitegenerators.net
- Metalsmith

建议:

[1] 在使用静态网页构建工具之前，请考虑使用 Gulp 来编排自定义解决方案，或者使用一个利用 Gulp 来生产静态网页的工具 例如. Gulp Starter

无障碍访问工具

指南

- 无障碍访问指南清单
- 互动 Web 内容无障碍访问指南（WCAG） 2.0
- 18F 无障碍访问指南

站点扫描工具

- aXe 浏览器扩展工具
- Chrome 无障碍访问开发者工具
- Tenon 无障碍访问工具
- WAVE 无障碍访问工具

颜色对比度检查工具

- Colorable
- Colorable Matrix
- Color Safe
- Color Ratio

视觉障碍模拟工具

- SEE (Chrome)
- Spectrum (Chrome)
- NoCoffee (Chrome)

屏幕阅读器

- VoiceOver (Mac)
- JAWS (Win)
- NVDA (Win)
- Window-Eyes (Win)
- ChromeVox (Chrome extension)

- 基础屏幕阅读指令

可读性检查工具

- Expresso App
- Hemingway App
- Grammarly
- Readability Score
- MS Office

相关文章

- ARIA 入门
- 重构 Web 的可访问性
- 无障碍访问入门
- 实用 ARIA 实例
- MDN 无障碍访问指南
- 在 Chrome 开发者工具中打开无障碍访问面板

应用程序框架工具（台式机，手机，平板电脑等）

前端应用程序框架：【^1】

- AngularJS （例如：Angular 1.x.x）+ Batarang
- Angular （例如：Angular 2.0.0 +）+ angular-cli
- Aurelia + Aurelia CLI
- Ember + embercli + Ember Inspector
- Polymer
- React + create-react-app + React开发工具
- Vue.js + vue-cli & Vue.js devtools
- Riot

原生混合移动**WebView**框架（例如：浏览器引擎驱动）：

这些解决方案一般使用Cordova、 crosswalk、或自定义WebView作为使用原生接口的桥梁。

- ionic
- onsen.io

原生混合移动开发**Webview**环境**/**平台**/**工具 （例如：浏览器引擎驱动）**:**

这些解决方案一般使用Cordova、 crosswalk、或自定义WebView作为使用原生接口的桥梁。

- Adobe PhoneGap [$]
- AppBuilder [$]
- cocoon.io [免费$]
- ionic hub [免费$]
- kony [$]
- Monaca [$]
- Taco

原生桌面**WebView**应用程序框架**:**（例如：浏览器引擎驱动）**:**

- Electron
- NW.js

各种平台应用程序框架：

这些解决方案可以让您在多个平台和设备上构建应用程序。

- manifoldJS

原生移动应用程序框架（又叫做原生**JavaScript**应用程序）

这些解决方案在运行时使用JS引擎解析JS并连接到原生接口。没有使用浏览器引擎和WebView。 UI由本机UI组件构成。

- NativeScript
- React Native
- tabris.js [免费$]
- trigger.io [$]
- weex

参考：

- todomvc.com
- Frontend Guidelines Questionnaire
- Frontend Guidelines

性能：

- js-framework-benchmark

注意：

2017年基于UI应用程序构建组件时需要注意inferno、Svelte、 和NX。

建议：

1 如果你是前端/JavaScript应用程序开发的新手，建议从Riot 或 Vue.js开始着手。其次学习React。然后Angular 2、 Ember、或者 Aurelia。

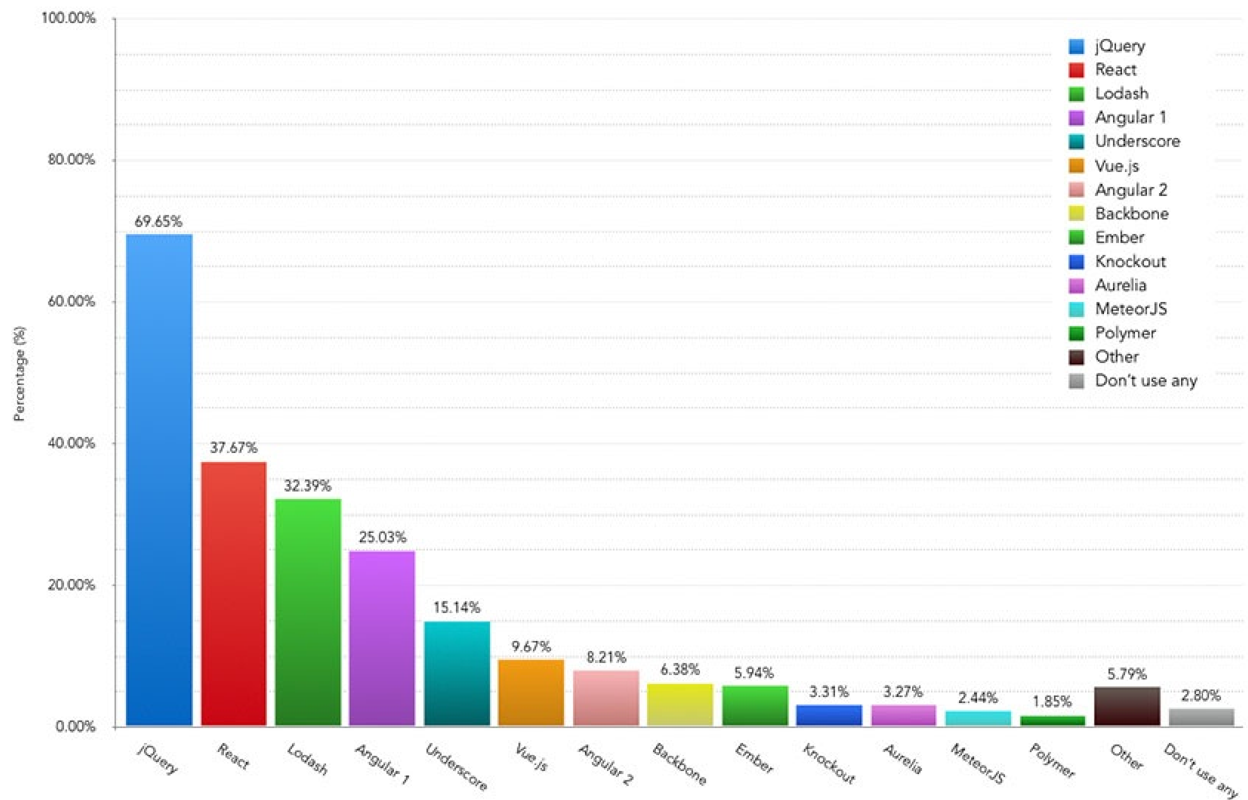
如果你开发的是一个数据交互少的小型网站（基本上是一个静态页面），这时不需要使用前端框架。许多工作可以使用任务运行工具如Gulp 和 jQuery来替代，同时可以避免不必要的学习和使用复杂的应用程序框架工具。

想要比React更小的框架，可以考虑下Preact。Preact是对React重构核心的议案（类似像Mithril的库）， 使用尽可能少的代码，并对ES2015提供一流的支持。目前库的大小是3kb左右（压缩和打包之后）。

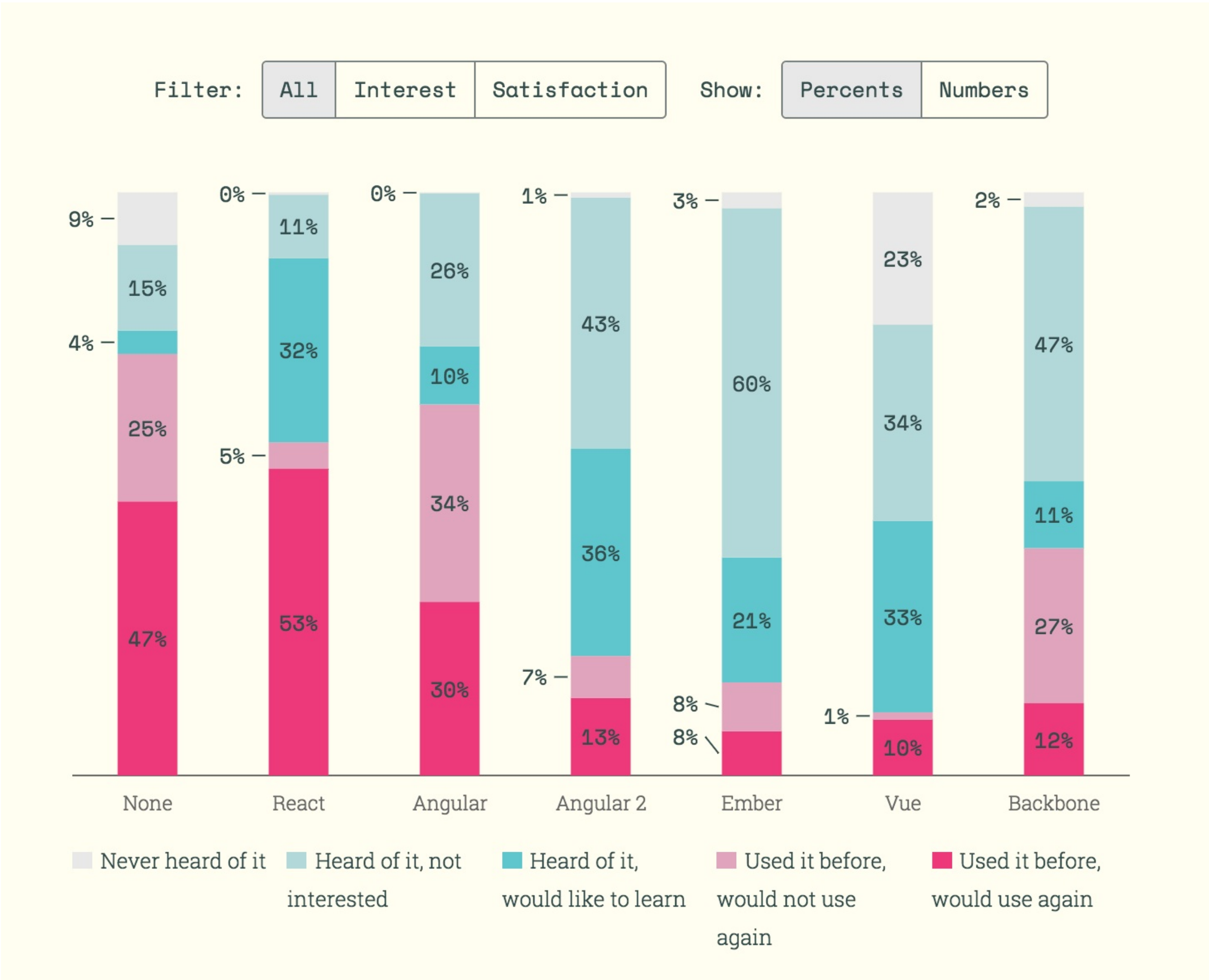
无法决定使用React还是Angluar 2时，请阅读，"Angular 2 vs React：终极对决"。

调查结果：

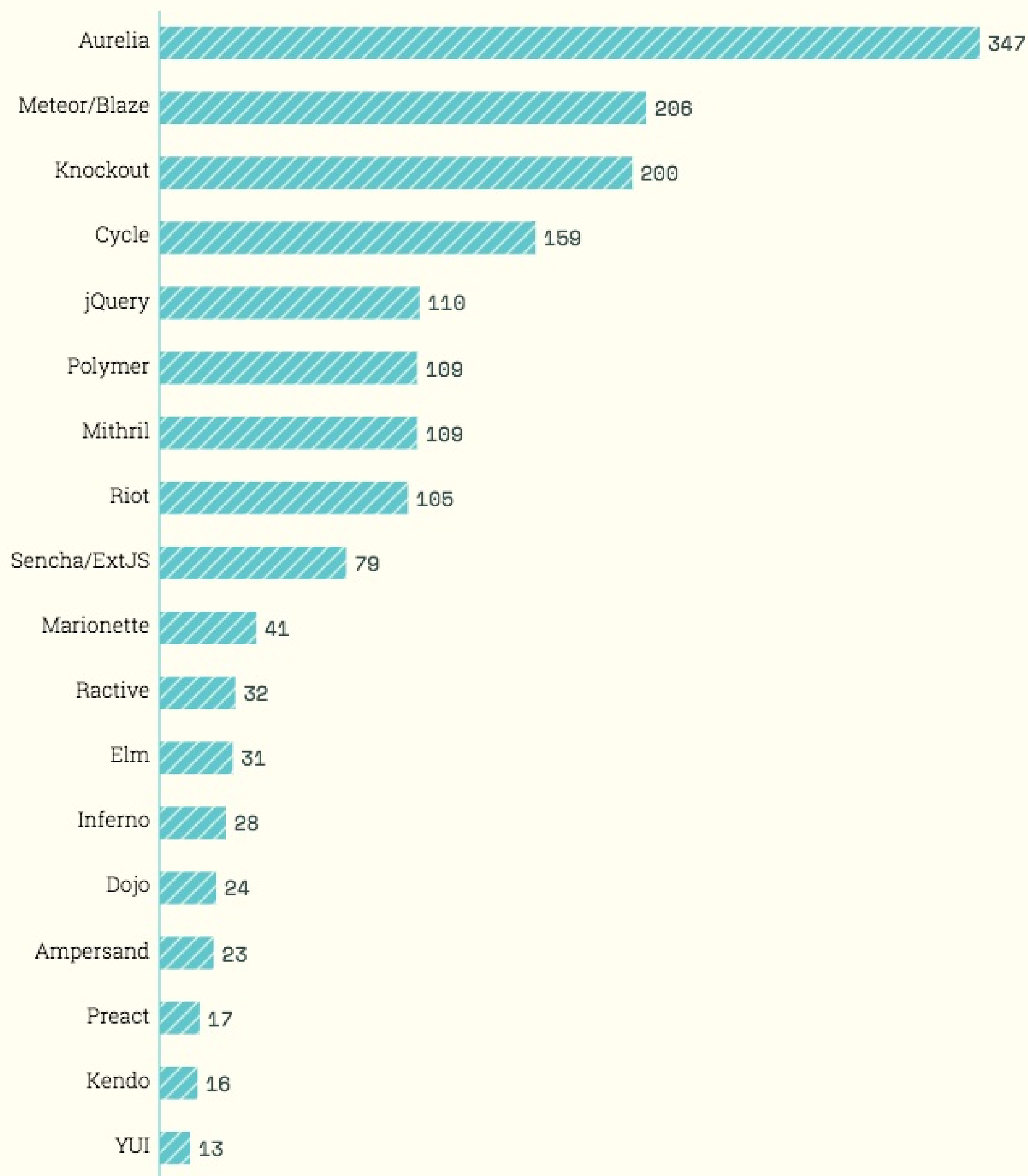
下面图片的来自 2016前端工具调查 (4715名开发者) 和 2016年JS调查状况（9307名开发者）



图片来源：*https://ashleynolan.co.uk/blog/frontend-tooling-survey-2016-results*



图片来源： http://stateofjs.com/



图片来源：*http://stateofjs.com/*

渐进式 Web 应用工具：

前端 **APP** 开发框架**:**

- 灯塔（lighthouse）
- 渐进式 Web 应用程序开发清单

脚手架工具

客户端脚手架是为了给应用搭建一个整体性的，从无到有的开发模版，而不是 构建可以访问数据库的代码 。

- Slush
- Yeoman

常规前端开发工具

开发工具:

- Browsersync
- CodeKit
- Prepros

模版／数据绑定工具

模版:

- doT.js
- Handlebars
    - htmlbars
- Nunjuncks

模版和动态数据绑定:

- Deku
- jquerymy.js
- ractive.js
- react.js
- riot
- Rivets.js
- vue.js

用于虚拟 **DOM** 的模版:

- JSX
- t7

UI组件 & 组件包

在 **Web** 平台的**:** [1]

- Bootstrap 3 or Bootstrap 4
- Kendo UI for jQuery [free to $]
- Materialize
- Office UI Fabric
- Semantic UI
- UiKit
- Webix [$]

React 针对 **Web** 平台的**:** [2]

- Ant Design
- Material ui
- Semantic-UI-React

通过 **Web** 平台进行 **native** 开发的桌面／笔电／小笔电应用的 **(**例如，和 **NW.js**、**Electron** 一起使用**):**

- Photon
- React UI Components for OS X El Capitan and Windows 10

移动端／平板针对 **Web** 平台的**(**例如，与带有交互的触摸一起用**):**

- Framework7
- Kendo UI Mobile
- Ratchet

建议：

[1] 如果你需要一套基础的 UI 组件，开始用Semantic UI吧。如果你正构建一个带有栅格、计算表或透视网格的应用，看看Kendo UI 或 Webix吧。并且要记住，大多数的这些解决方案仍需要 jQuery。

[2] 如果我准备构建一个 React 应用，需要用一个现成的组件工具包，我更倾向于用Semantic-UI-React 和／或Ant Design，而且不得不得接受这一事实：我想用的一些组件和使用对 jQuery 有很强的依赖。

数据可视化工具（例如，图表）

JS 代码库:

- d3
- sigmajs

小工具和组件:

- amCharts [免费增值]
- AnyChart [非商业性质 免费增值]
- C3.js
- Chartist-jsj
- Chart.js
- Epoch
- FusionCharts [付费]
- Google Charts
- Highcharts [非商业性质 免费增值]
- ZingChart [免费增值]

服务 (例如，嵌入和分享（数据）的托管数据可视化服务):

- ChartBlocks [免费增值]
- Datawrapper
- infogr.am [免费增值]
- plotly [免费增值]

图形工具（例如 SVG、canvas、webGL）

通用工具:

- Fabric.js
- Two.js

Canvas:

- EaselJS
- Paper.js

SVG:

- d3
- GraphicsJS
- Raphaël
- Snap.svg
- svg.js

WebGL:

- pixi.js
- three.js

动画工具

- Animate
- Anime
- Animista.net
- Dynamics.js
- GreenSock-JS
- Magic
- TweenJS
- Velocity.js

Polyfills/Shims:

- web-animations-js

动画参考资料:

- canianimate.com

JSON 工具

在线编辑工具:

- JSONmate
- json.browse()

格式化工具和校验器:

- jsonformatter.org
- JSON 格式化工具和校验器

查询工具:

- DefiantJS
- JSON Mask
- ObjectPath

模拟生成 **JSON** 数据的工具:

- JSON Generator
- Mockaroo [增值付费]

在线 **JSON** 模拟 **API** 工具:

- FillText.com
- Jam API
- JSONPlaceholder
- jsonbin.org
- mockable.io
- mockapi.io
- Mocky
- RANDOM USER GENERATOR

公共 **JSON API** 列表:

- 开发中使用的 JSON APIs 列表

本地 **JSON** 模拟 **API** 工具:

- json-server

JSON 规范:

- json-schema.org & jsonschema.net
- {json:api}

内容占位符工具

图片：

- placehold.it
- Satyr
- Placeimg
- Lorem Pixel
- CSS-Tricks Image Resources
- LibreStock
- Unsplash
- Place Beyoncé

设备模拟：

- placeit.net
- mockuphone.com

文本：

- Meet the Ipsums
- catipsum.com
- baconipsum.com (API)

用户数据：

- uinames.com
- randomuser.me

测试工具

软件测试框架：

- Intern
- Karma
- Jest

单元测试：

- AVA
- Jasmine
- Mocha
- Tape

单元测试的使用的断言工具：

- Chai
- expect.js
- should.js

单元测试的 **Test Spies**、**Stubs** 和 **Mocks**：

- sinon.js
- Kakapo.js

浏览器托管测试/自动化测试：

- Browserling [$]
- BrowserStack [$]
- CrossBrowserTesting.com [$]
- Nightcloud.io
- Sauce Labs [$]

浏览器自动化：

- CasperJS
- Nightmare
- TestCafe

UI 测试工具：

- gremlins.js

- Percy
- BackstopJS
- PhantomCSS
- Ghost Inspector
- diff.io

死链和错误自动检测器：

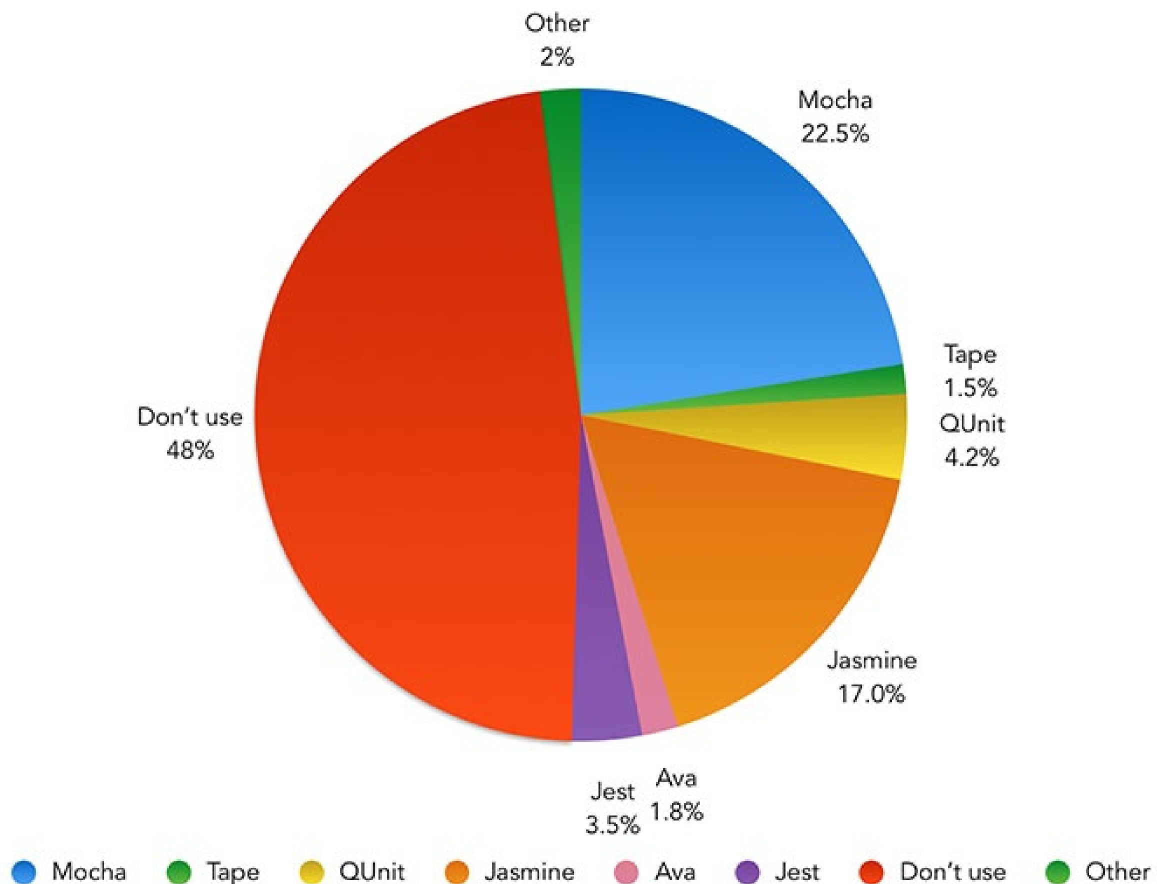
- Monkey Test It

备注:

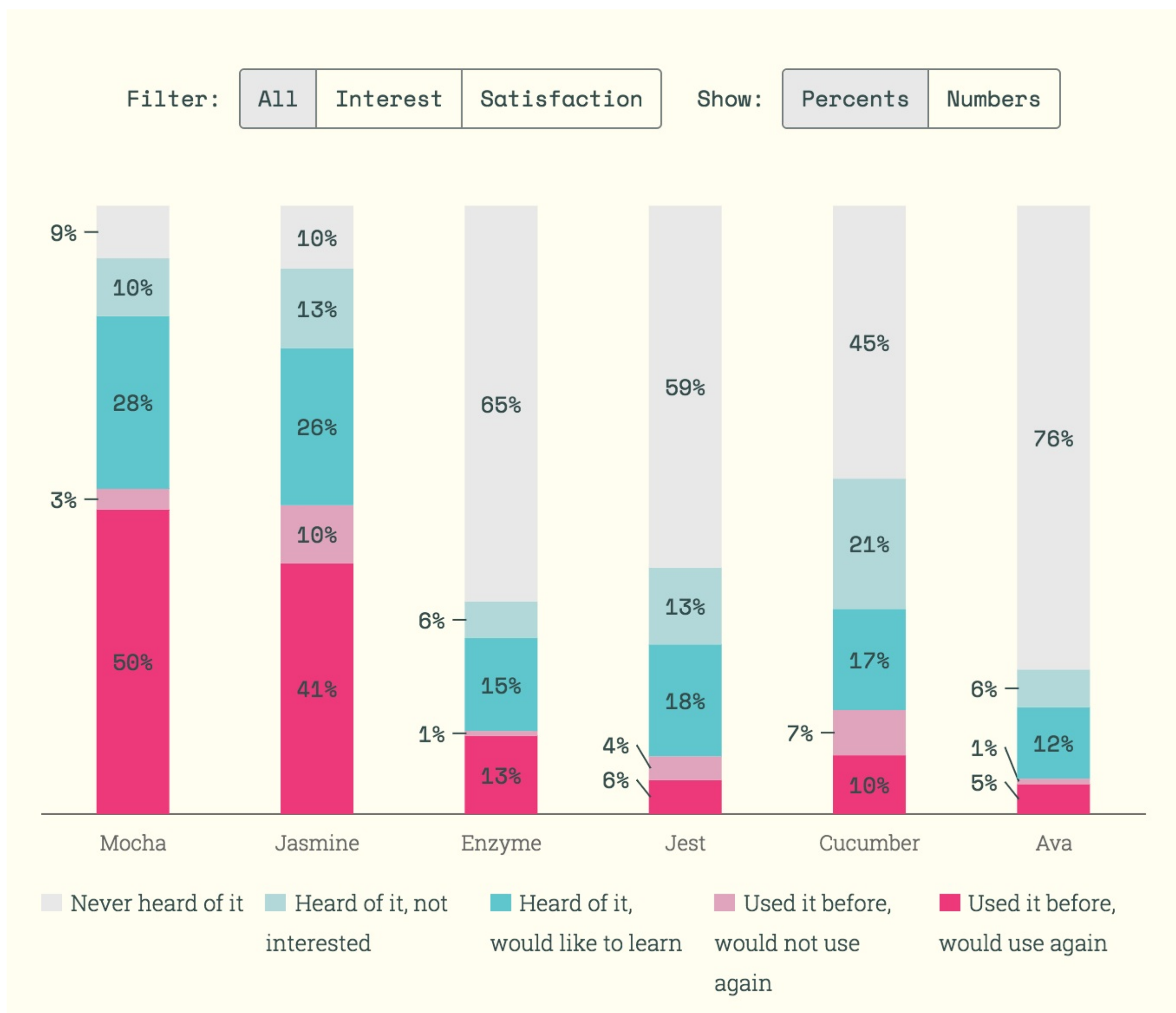
测试框架通常提供了更多的工具，而不仅仅是单元测试。如果您正在寻找 JavaScript 单元测试解决方案，请查看 JavaScript Tools。

调查结果：

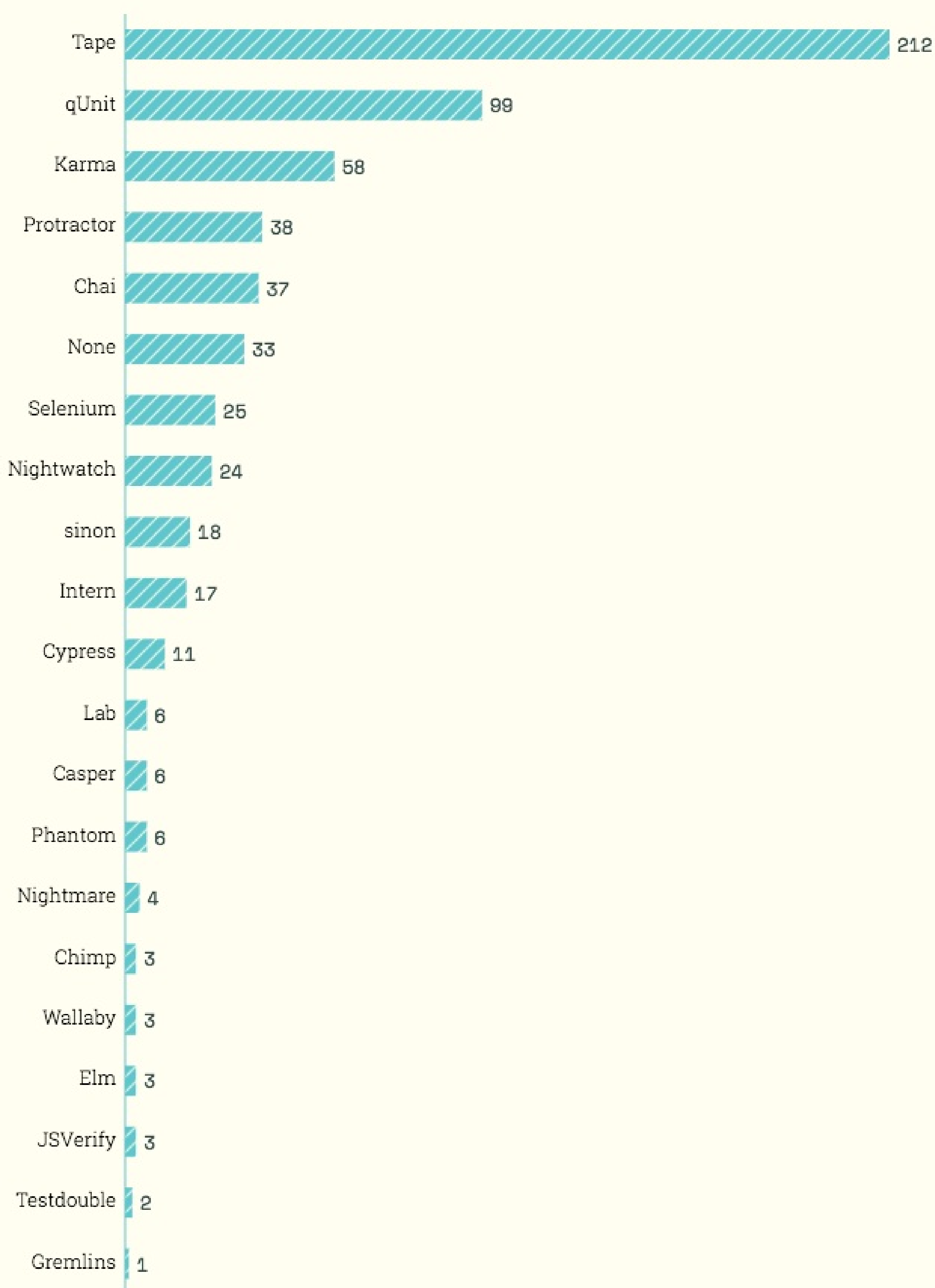
以下图片来自 2016 年前端工具调查（4715 位开发人员）和 2016 年度 JS 调查报告（9307 位开发人员)：



图片来源：*https://ashleynolan.co.uk/blog/frontend-tooling-survey-2016-results*



图片来源：*http://stateofjs.com/*



图片来源：http://stateofjs.com/

前端数据存储工具 (例如. 客户端的数据存储方案)

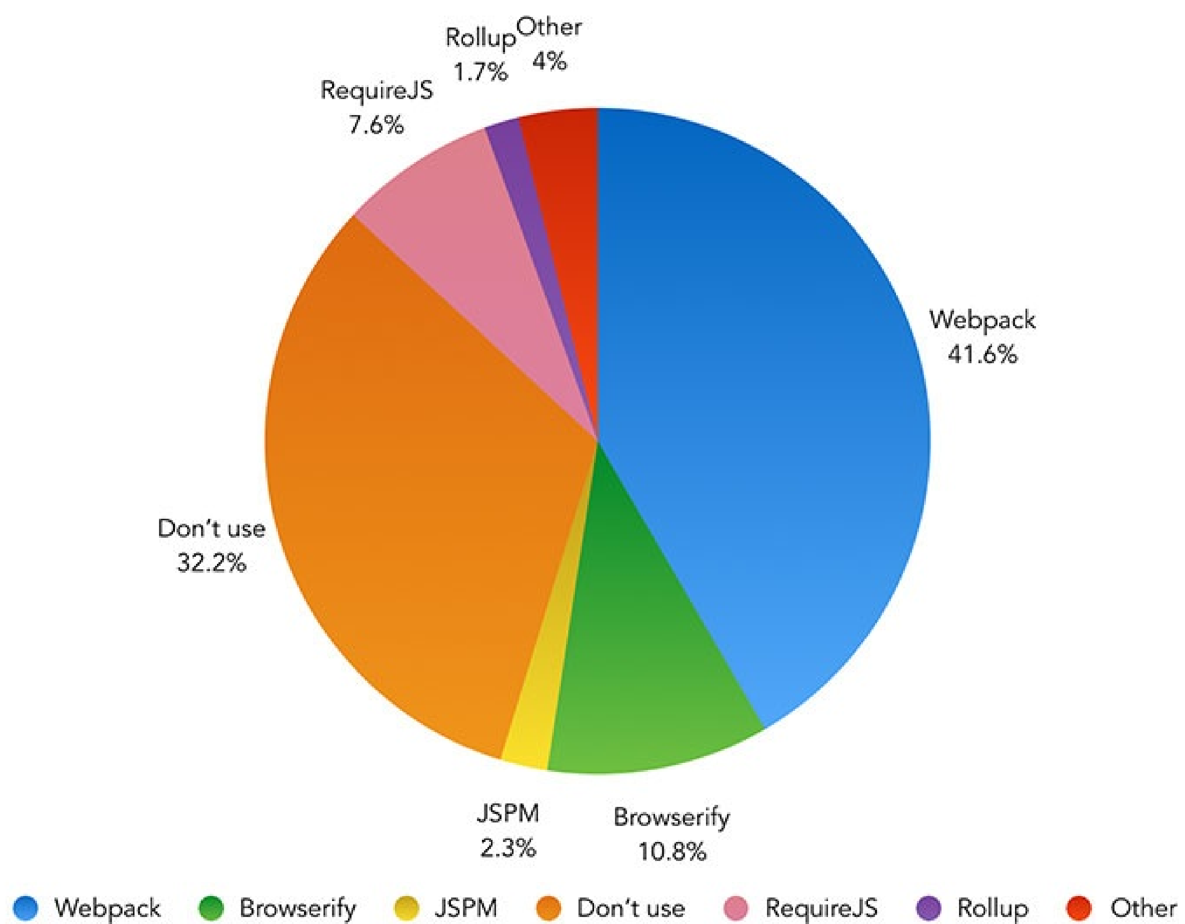
- AlaSQL
- Dexie.js
- ForerunnerDB
- LocalForage
- LokiJS
- Lovefield
- lowdb
- Pouchdb
- NeDB
- YDN-DB

模块加载／打包工具

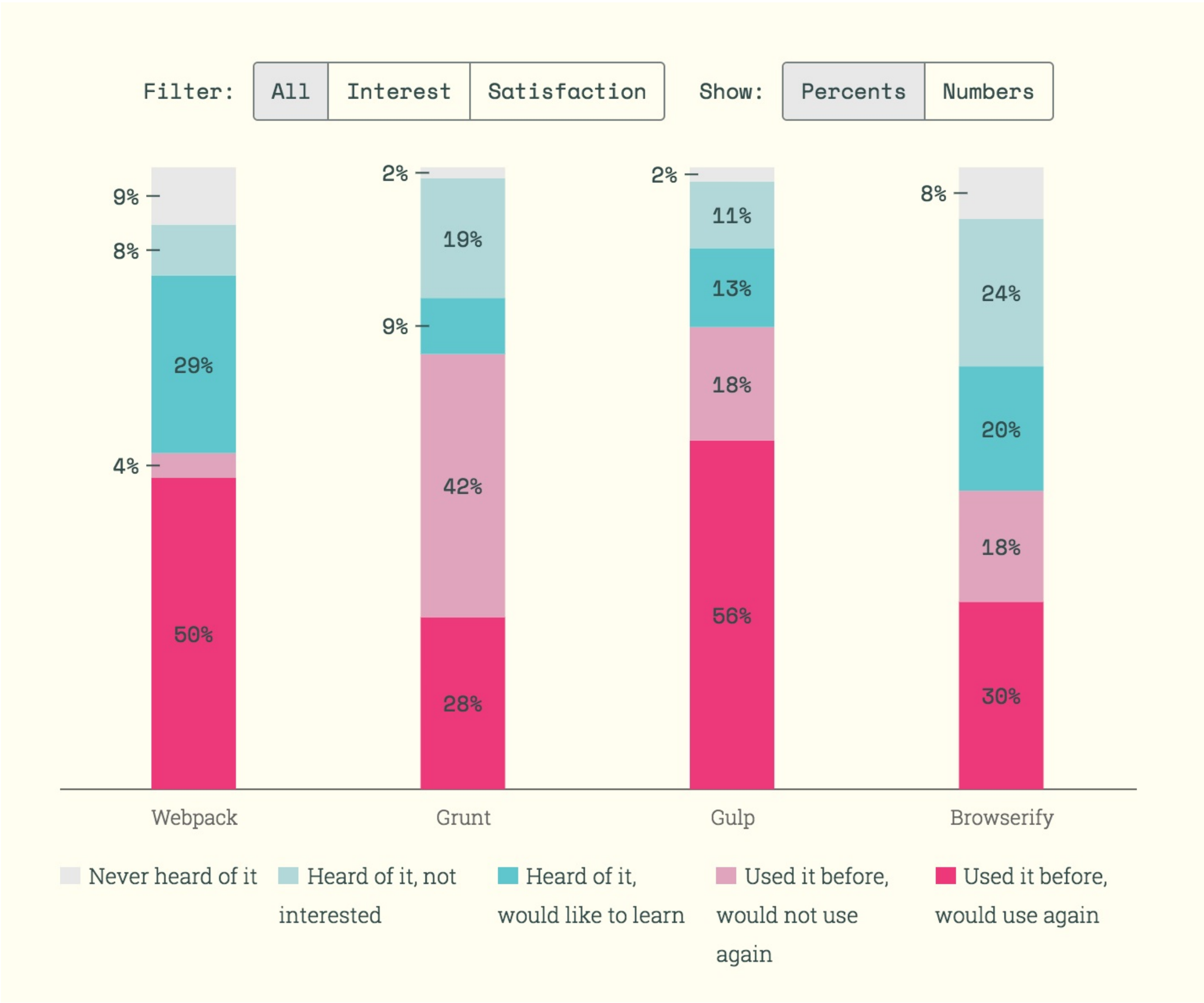
- Browserify
- Rollup
- SystemJS
- webpack
  - http://www.webpackbin.com/

调查结果：

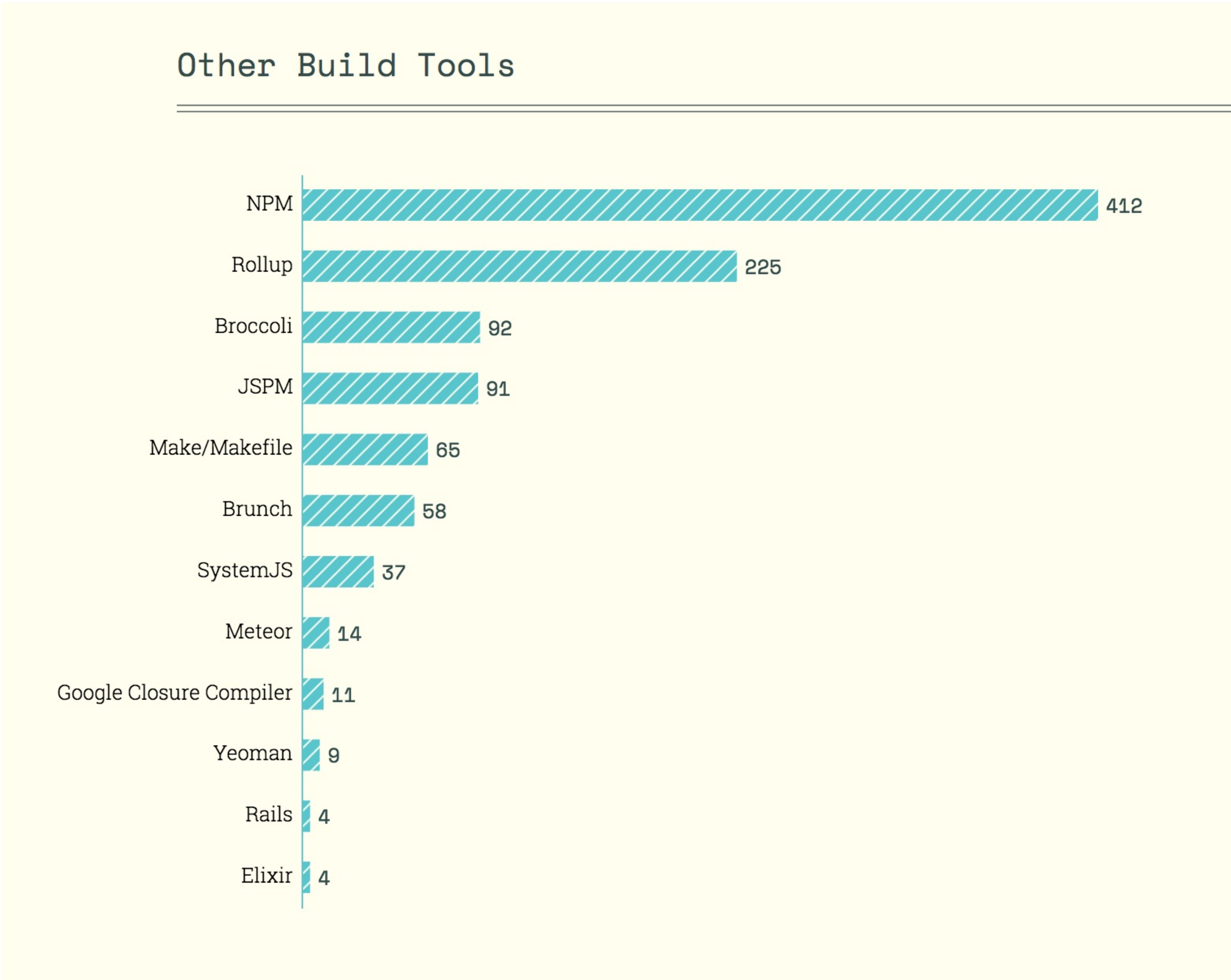
下图是来自 2016 前端工具调查 (4715 名开发者参与) 以及 2016 JS 现状调查 (9307 名开发者参与)



图片来源：*https://ashleynolan.co.uk/blog/frontend-tooling-survey-2016-results*



图片来源: http://stateofjs.com/



图片来源: http://stateofjs.com/

模块／包管理工具

- NPM
- yarn

托管工具

通用工具

- AWS [付费]
- DigitalOcean [付费]
- Heroku [免费增值]

静态工具

- Firebase Hosting
- netlify [免费增值]
    - Bitballoon
- Surge [免费增值]
- Forge [付费]

项目管理 以及 代码托管工具

- Assembla [免费增值]
- Bitbucket [免费增值]
- Codebase [付费]
- Github [免费增值]
- GitLab [免费增值]
- Unfuddle [付费]

协作与沟通工具

- Slack & screenhero [免费增值]
- appear.in
- Mattermost [免费增值]
- TeamViewer [免费增值]

代码／**GitHub** 协作与沟通**:**

- Gitter [免费增值]

内容管理 托管／API 工具

API 内容管理系统 **(i.e.,** 内容分发 **CMS)** 工具：

- Contentful [付费]
- Cosmic JS [免费增值]
- prismic.io [免费增值]
- elemeno [免费增值]

托管 **CMS** 工具**:**

- Cushy CMS [免费增值]
- LightCMS [付费]
- Page Lime [付费]
- Surreal CMS [付费]

静态 **CMS** 工具**:**

- webhook.com
- Dato CMS
- siteleaf
- forestry.io

后端／API 工具

数据／后端即服务（**BAAS**）：

- Back& [免费增值]
- Firebase [免费增值]
- Kinvey [免费增值]
- Pusher [免费增值]
- restdb.io [免费增值]

数据／后端

- Horizon
- GraphQL
    - http://www.apollodata.com/
    - Relay
- Falcor
- RxDB

用户管理作为一项服务：

- Auth0 [付费]
- AuthRocket
- Stormpath
- UserApp [免费增值]

离线工具

- Hoodie
- Offline.js
- PouchDB
- upup

安全工具

编码工具：

- DOMPurify
- XSS

安全性 扫描工具/评估工具/测试工具：

- Netsparker
- Websecurify
- OWASP ZAP

参考文献：

- HTML5 安全秘籍

构建工具

任务管理 / 构建工具: [1]

- Gulp
- Broccoli.js

自定义任务管理／构建管道工具:

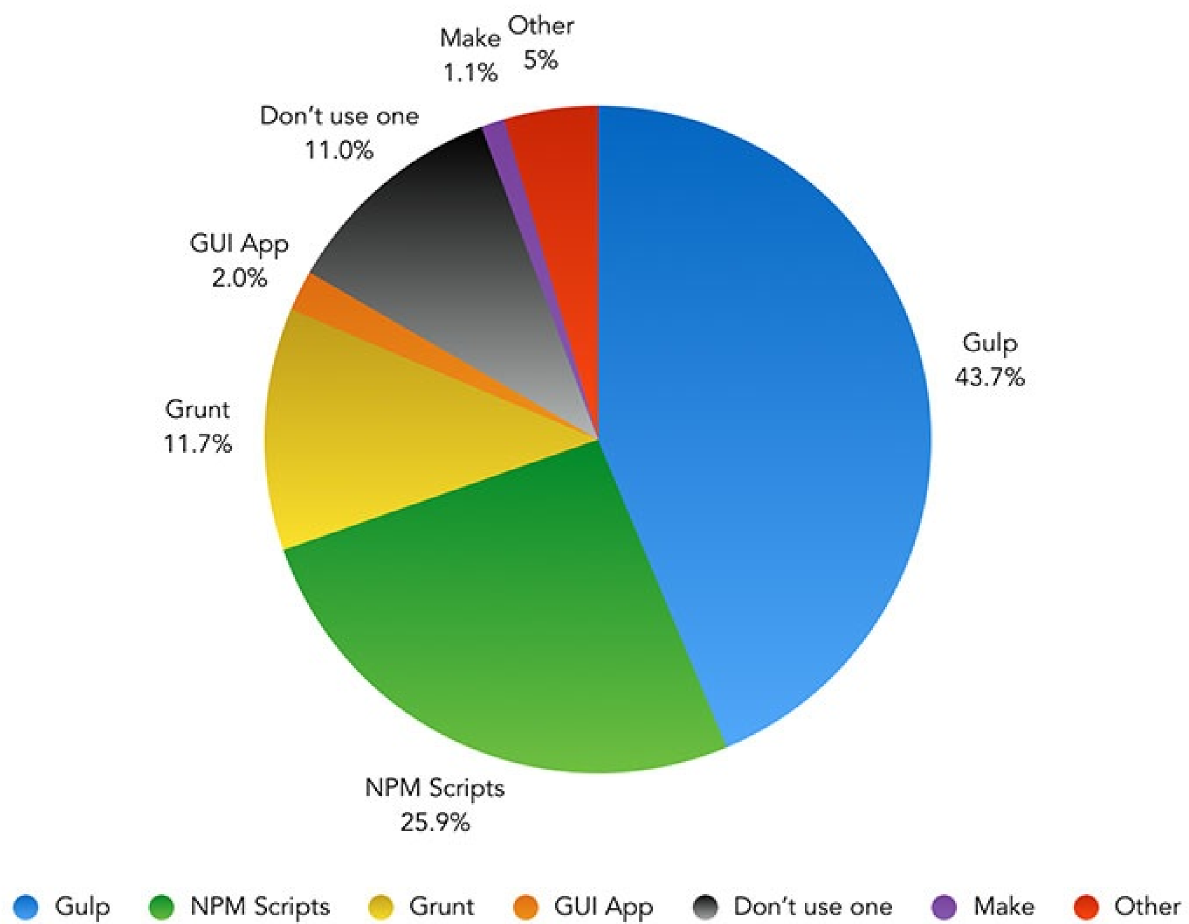
- Brunch
- Mimosa
- Lineman

建议:

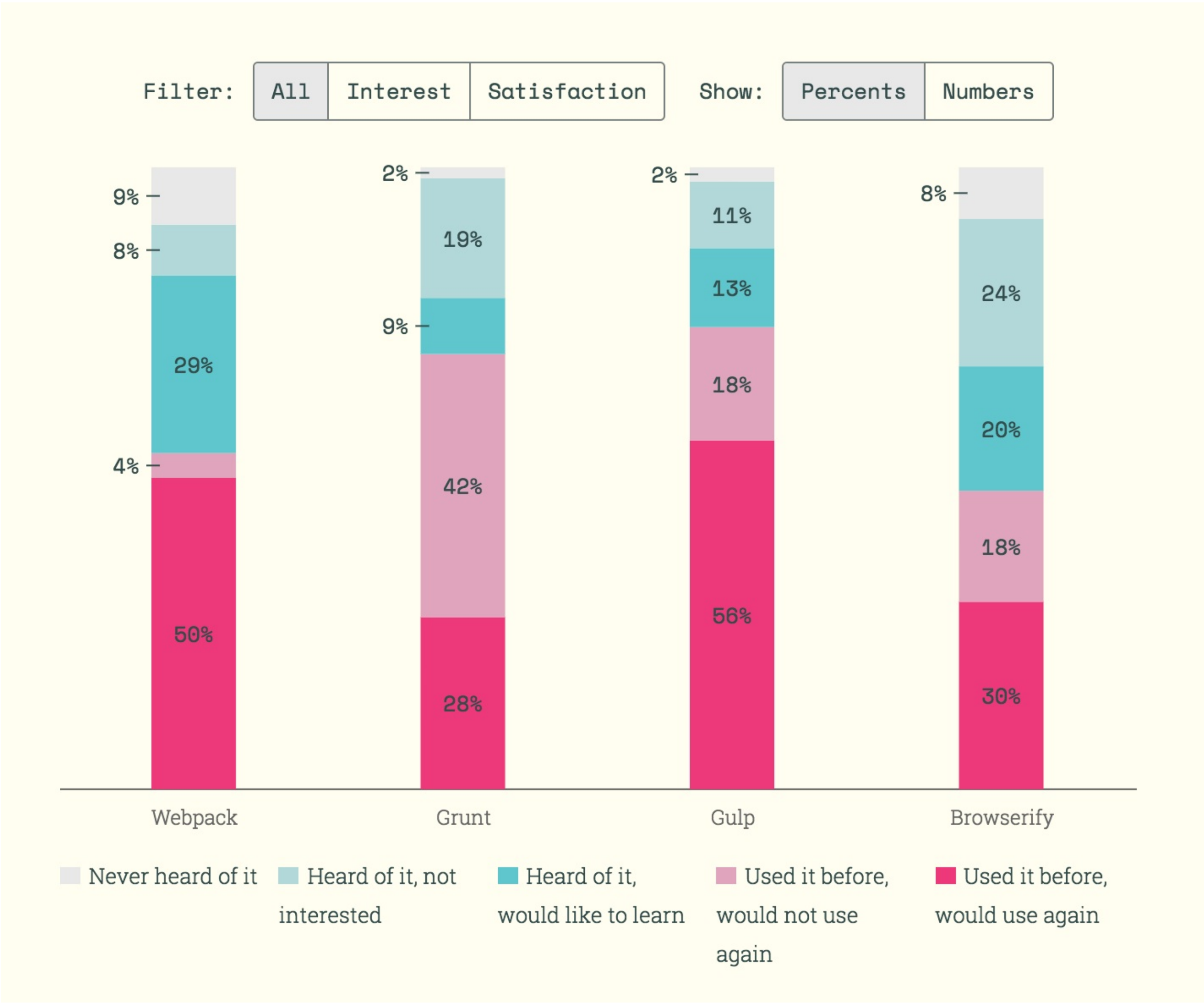
[1] 在决定采用Gulp之前，请确认npm scripts 或者 yarn script 不能满足开发要求。请参阅"我为什么用 npm Scripts 代替 Gulp 和 Grunt".

调查结果:

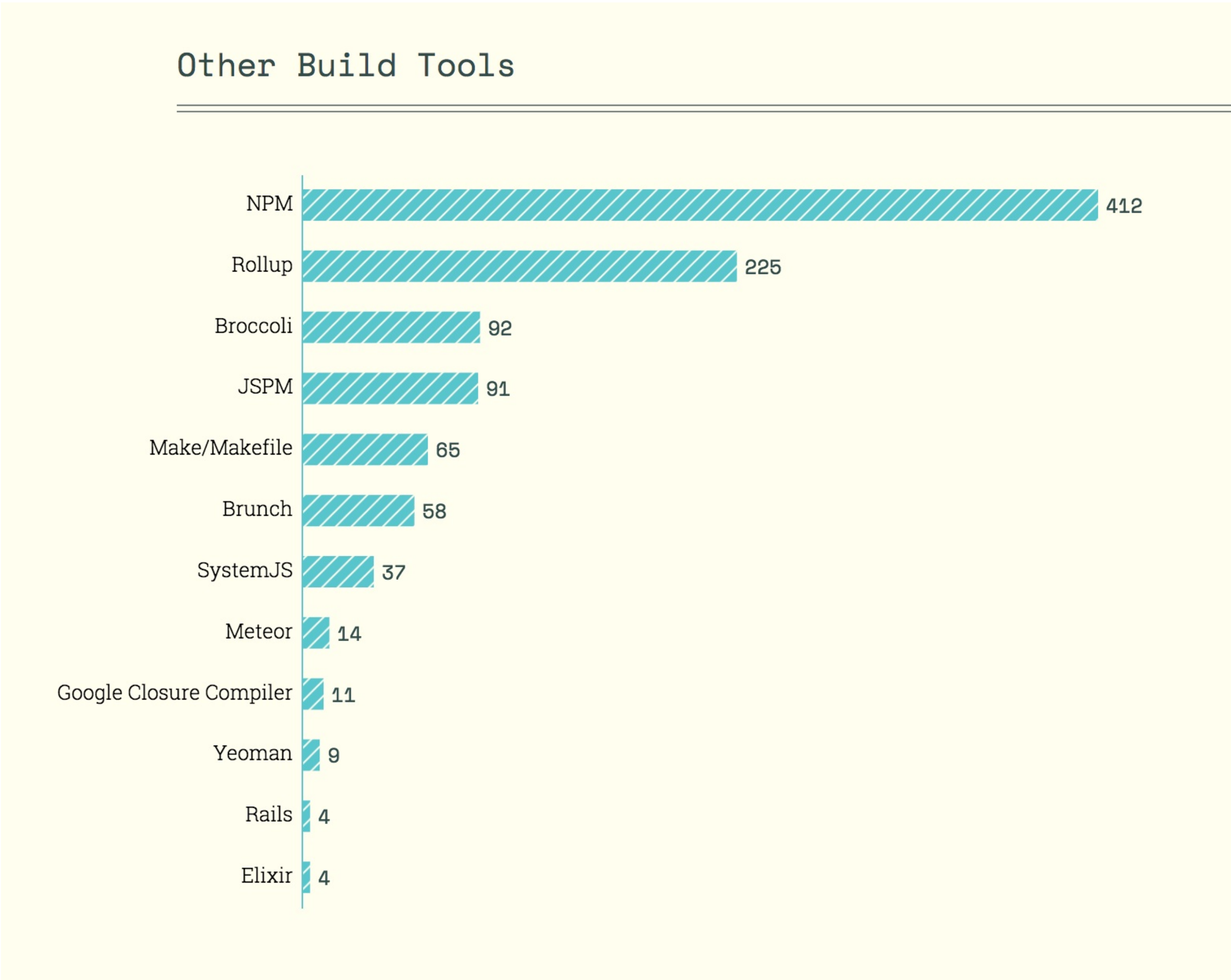
下图来自于 2016 前端工具调查 (4715 参与者) 以及 2016 JS 使用情形调查 (9307 参与者)



图片来源: *https://ashleynolan.co.uk/blog/frontend-tooling-survey-2016-results*



图片来源: *http://stateofjs.com/*



图片来源: *http://stateofjs.com/*

部署工具

- Bamboo [付费]
- Buddy [免费增值]
- CircleCI [免费增值]
- Codeship [免费增值]
- Deploybot [免费增值]
- Deployhq [免费增值]
- FTPLOY [免费增值]
- Now [免费增值]
- Travis CI [免费增值]
- Semaphore [免费增值]
- Springloops [免费增值]

网站／应用监控工具

正常运行时间监测：

- Monitority [免费]
- Uptime Robot [免费增值]

整体监测工具：

- Pingdom [免费增值]
- New Relic
- Uptrends [免费]

JavaScript 错误报告／监控

- bugsnag [付费]
- errorception [付费]
- Honeybadger [付费]
- Raygun [付费]
- Rollbar [免费增值]
- Sentry [免费增值]
- TrackJS [付费]

性能工具

报告:

- GTmetrix
- sitespeed.io
- Speed Curve [$]
- Web Page Test

JS 工具:

- imagemin
- ImageOptim-CLI

预算:

- performancebudget.io

参考文献:

- Jank Free
- ES6 与 ES5 性能比较

查阅表格:

- 2017 前端性能查阅表

寻找工具的工具

- built with
- javascripting.com
- js.coach
- microjs.com
- npms
- stackshare.io
- Unheap